Computational Science Requirements for Leadership Computing

July 2007

National Center for Computational Sciences
Leadership Computing Facility • Oak Ridge National Laboratory

ORNL/TM-2007/44

Leadership Computing Facility
National Center for Computational Sciences
Oak Ridge National Laboratory

COMPUTATIONAL SCIENCE REQUIREMENTS FOR LEADERSHIP COMPUTING

Douglas Kothe
Ricky Kendall

Date Published: July 2007

Prepared by
OAK RIDGE NATIONAL LABORATORY
P.O. Box 2008
Oak Ridge, Tennessee 37831-6254
managed by
UT-Battelle, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

FIGURES

TABLES

ABBREVIATIONS AND ACRONYMS

AMR	adaptive mesh refinement
ANL	Argonne National Laboratory
ARMCI	aggregate remote memory copy interface
ARC	Application Requirements Council
ASCAC	Advanced Scientific Computing Advisory Committee
ASCR	Advanced Scientific Computing Research
BLAS	basic linear algebra subprograms
BNL	Brookhaven National Laboratory
CAF	Co-Array Fortran
CC	coupled cluster
CCSM	Community Climate System Model
CET	center for enabling technologies
CFD	computational fluid dynamics
CG	conjugate gradient
CIOD	control and I/O daemon
CTEM	collisionless trapped electron mode
CY	calendar year
DFT	density functional theory
DNA	deoxyribonucleic acid
DOE	Department of Energy
DOF	degrees of freedom
DWF	domain wall fermion
ESSL	Engineering Scientific Subroutine Library
FEM	finite element method
FPGA	field-programmable gate array
FFT	fast Fourier transform
FY	fiscal year
GB	gigabyte or 10^9 bytes
HDF	hierarchical data format
HPC	High Performance Computing
I/O	input/output
IDE	integrated development environment

ILC	International Linear Collider
INCITE	Innovative and Novel Computational Impact on Theory and Experiment
INL	Idaho National Laboratory
ITER	International Thermonuclear Experimental Reactor
ITG	ion temperature gradient
Jaguar	Cray XT supercomputer at ORNL NCCS
KKR	Korringa-Kohn-Rostoker
LANL	Los Alamos National Laboratory
LAPACK	linear algebra package
LC	Leadership Computing
LCF	Leadership Computing Facility
LHC	Large Hadron Collider
LHPC	Large Hadron Physics Collider
LOC	Lines of Code
MB	megabyte or 10^6 bytes
MD	molecular dynamics
MOC	meridianal overturning circulation
MPI	message-passing interface
MP2	mathematics-physics platform
MSP	multistreaming vector processor
MTTI	mean time to interrupt
NCCS	National Center for Computational Sciences
NERSC	National Energy Research Scientific Computing Center
netCDF	network common data form
NNSA	National Nuclear Security Agency
NSF	National Science Foundation
NSLER	nuclear structure and low energy reactions
NUCCOR	Nuclear Coupled Cluster — Oak Ridge
ODE	ordinary differential equation
ORNL	Oak Ridge National Laboratory
PB	Petabyte or 10^{15} bytes
PDE	partial differential equation
PETSc	portable, extensible toolkit for scientific computation
PF	Petaflops or 10^{15} floating-point operations per second

PI	principal investigator
Phoenix	Cray X1E supercomputer at ORNL NCCS
PPPL	Princeton Plasma Physics Laboratory
QCD	quantum chromodynamics
QMC	Quantum Monte Carlo
QMD	quantum molecular dynamics
RAM	random access memory
RBC	RIKEN/BNL/Columbia University
RG	renormalization group
RHMC	Rational Hybrid Monte Carlo
SAP	Scientific Application Partnership
SC	Office of Science
SciDAC	Scientific Discovery through Advanced Computing
SLAC	Stanford Linear Accelerator Center
SMP	Symmetric Multi-Processor
SPRNG	Scalable Parallel Pseudo Random Number Generator
SUSY	supersymmetry
TB	terabyte or 10^{12} bytes
TC	Technology Council
TF	teraflops or 10^{12} floating-point operations per second
WAN	wide area network
XML	extensible markup language

EXECUTIVE SUMMARY

In 2006 the Department of Energy (DOE) Leadership Computing Facility (LCF) at the Oak Ridge National Laboratory (ORNL) National Center for Computational Sciences (NCCS) elicited petascale computational science requirements from leading computational scientists in the international science community. Targeted were those scientific teams whose projects were recipients of large computer allocation awards on DOE LCF systems (such as the ORNL NCCS) and National Science Foundation (NSF) Centers. The overwhelming response from this distinguished group of scientists was a call for a balanced, well-integrated, and reliable system.

We found, not surprisingly, that each of approximate dozen principal LCF system attributes is interdependent upon one another. With scientific discovery as the principal objective, greatly increasing the potential of one particular attribute (e.g., peak flops) can and should only be done while simultaneously increasing other attributes (e.g., memory) having a "shared fate," thus the need for a "balanced" system. Each system attribute (e.g., peak flops, mean time to interrupt, network bandwidth, node memory, local storage, archival storage, memory latency, communication latency, disk latency, interconnect bandwidth, memory bandwidth, disk bandwidth) cannot be considered and optimized in isolation.

Requirements elication, analysis, validation, and management is a difficult and inexact process. It is especially difficult when reliable, quantitative extrapolations are sought. The results of this first annual leadership computational science requirements elication and analysis by the ORNL NCCS are more qualitative in nature (e.g., science achievable at the petascale), but important nevertheless. As this annual process continues, the analyses and extrapolations will become more quantitative and actionable. With that said, the analysis contained herein did lead to tangible, actionable decisions for the ORNL NCCS (e.g., required local and archival I/O bandwidth and capacity).

Based on interactions with leading scientists in the field, the LCF system attributes expected to have the greatest impact on existing and developing science applications have been identified. In the area of system hardware, the attributes found to be most critical to maximizing the potential for breakthrough science were peak flops, node memory capacity, interconnect latency, interconnect bandwidth, and memory bandwidth. Scientific applications of course impose requirements on the LCF systems, but the LCF systems, in fact, impose real requirements on the applications as well. For applications to execute efficiently on LCF systems, for example, they must possess algorithm and software attributes such as 10^5–10^6 task and hybrid (task/thread) parallelism, multilevel (in time and space) linear/nonlinear solution techniques, multiphysics coupling and time integration schemes, data structures that minimize bandwidth and maximize locality, and efficient parallel I/O.

We conclude with recommendations for the computing capabilities that scientists will need in future LCF systems. Scientific discoveries and breakthroughs cannot be "planned" with LCF systems, but by optimally matching these systems with the scientific needs and goals, discoveries and new levels of fundamental understanding are virtually guaranteed. The HPC and computational science communities now are truly at the forefront of a very exciting scientific renaissance.

INTRODUCTION

Refining requirements is a two-way street. The code developers must understand the petascale computer's capabilities and limitations, and HPC architects must understand the needs of a wide variety of science domain codes.

A requirement is a condition or capability needed by a user to solve a problem or achieve an objective. A requirement is also a condition or capability that must be met or possessed by a system to satisfy a contract, standard, specification, or other formally imposed document. Both definitions apply for the breakthrough computational science requirements used in the design, procurement, deployment, and operation of the Department of Energy (DOE) Leadership Computing Facility (LCF) at the National Center for Computational Sciences (NCCS). This document contains critical leadership computational science requirements for the key science areas of interest to the DOE, for improved science quality and productivity, for higher fidelity physical model and numerical algorithm requirements, for more efficient and higher quality software, and for in-depth data analytics and work flow.

By articulating these requirements and using them to manage and arbitrate decisions, the NCCS will align LCF systems to the maximum extent possible with the needs and goals of the breakthrough science projects using these resources. LCF requirements for the NCCS apply to the entire end-to-end analysis process that scientists follow when using the NCCS facilities. This process comprises system hardware, system software, the integrated development environment, and the problem-solving environment that includes data analysis, management, and visualization. We expect that effective requirements development, management, and planning will positively influence the design, procurement, deployment, and operation of an NCCS system by improving the quality, quantity, or fidelity of the output of one or more breakthrough science simulation applications in a measurable way. For requirements to be useful to the NCCS, they must be actionable and as quantitative as possible without being solutions themselves. In reality, requirements flow in both directions: applications impose requirements on the LCF systems, and the LCF systems in turn impose requirements upon the applications.

A valid requirements process must follow three basic steps: planning, development, and management. The NCCS requirements effort in 2006 was principally devoted to establishing the methods by which the three-step requirements process is executed, and to initiating the first step in the requirements process—requirements development. Elicitation is key in requirements development. It is the ongoing process of analyzing existing documentation (see refs. 1–7 at the end of this section) and interviewing stakeholders.

Information from stakeholders was elicited with three separate surveys that were derived and/or given directly to the science project team members. These surveys were

- A requirements survey constructed and collected by the Application Requirements Council (ARC) (see Appendix C);
- A code project survey constructed by the ASCAC subpanel on science metrics for the Advanced Scientific Computing Research (ASCR) computing facility metrics (see Appendix D); and
- Answers to science and code questions from LCF and Innovative and Novel Computational Impact on Theory and Experiment (INCITE) project proposal applications (see Appendix E).

For the 22 projects allocated on the NCCS LCF systems in 2006, 8 projects responded to the ARC survey, 19 responded to the ASCAC survey, and all 22 filled out proposal applications (necessary for allocation awards). Answers to these surveys helped to define requirements from the following points of reference: science motivation and impact, science quality and productivity, application models, application algorithms, application software, application footprint, and data management and analysis. Requirements in general fall into four categories:

- Business ("why") requirements reflect the goals and objectives of the organization. Sources include project sponsors and key clients. Business requirements are described in project charters and vision statements.
- Functional ("what") requirements dictate what the product must do. Sources include end users, customers, regulations, and internal brainstorming. Functional requirements are found in use cases, specifications, interview notes, and models.
- Quality ("how well") requirements define the properties that the product must have: look and feel, usability, performance, operational environment, maintainability and portability, security, etc. Sources include end users, standards, and support teams. Quality requirements are found in models, specifications, use cases, and notes.
- Design ("how") requirements depict imposed design choices. Design requirements are found in models, specifications, and high-level designs.

In a broad sense, science motivation and impact reflect business requirements; science quality and productivity reflect quality requirements; application models reflect functional requirements; and algorithm, software, application footprint, and data analysis/management reflect design requirements. The elicitation process yields a series of documented responses that represent potential LCF requirements. These responses must then be analyzed and validated to ensure that they will result in workable requirements. Good requirements must be unambiguous, testable, correct, in scope, modifiable, feasible, traceable, written in clear (customer's) language, acceptable to all clients, and not themselves a solution.

SCIENCE DRIVERS

Over the past 5 years, DOE's Office of Science (SC), Scientific Discovery through Advanced Computing (SciDAC) Program, has achieved simulation-based scientific accomplishments through focused collaboration and active partnership of domain scientists, applied mathematicians, and computer scientists. The LCF at NCCS has played a role in many of these successes (e.g., nanoscience, accelerator design, astrophysics, chemistry, combustion, climate modeling, and fusion). Even more compelling opportunities for scientific discovery have fostered the new SciDAC-2 Program in which a series of coordinated investments across all DOE/SC Programs (Basic Energy Sciences, Biological and Environment Research, Fusion Energy Sciences, High-Energy Physics, and Nuclear Physics) promises to further the achievement of breakthrough science through (1) focusing efforts on scientific applications in specific domains and (2) enabling technologies in computer science, software infrastructure, and applied mathematics through centers for enabling technologies (CETs), university-led institutes, and scientific application partnerships (SAPs). SciDAC-2 thrust areas (with examples) include accelerator science [International Linear Collider (ILC) design], astrophysics (understanding of nucleosynthesis), climate modeling (global carbon cycle prediction), biology (protein interaction networks), fusion [International Thermonuclear Experimental Reactor (ITER) design], groundwater (subsurface reactive transport), high energy physics (dark universe and neutrinos), nuclear physics [National Nuclear Security Agency (NNSA) physics], and quantum chromodynamics (QCD) (lattice gauge theory). Other science areas ripe for discovery include nanoscience, chemistry, nuclear energy, and manufacturing (Table 1).

Table 1. Science drivers projects receiving a 2006 allocation on LCF systems at the NCCS

Science domain	Example science driver
Accelerator physics	Evaluate and optimize a new low-loss cavity design for the International Linear Collider (ILC) that has a lower operating cost and higher performance than existing designs.
Astrophysics	Determine the explosion mechanism of core-collapse supernova, one of the universe's most important sites for nucleosynthesis and galactic chemical enrichment. Determine details of the explosion mechanism of Type Ia supernova (thermonuclear explosions of white dwarf stars), helping to determine key characteristics for their use as standard candles for cosmology.
Biology	Help address the current oil and gasoline crisis by studying the ethanol option, including the most efficient means of converting cellulose to ethanol.
Chemistry	Study the catalytic transformation of hydrocarbons, clean energy and hydrogen production and storage, and the chemistry of transition metal clusters, including metal oxide.
Climate	Focus on the Grand Challenge of climate change science: predict future climates based on scenarios of anthropogenic emissions and other changes resulting from options in energy policies. Simulate the dynamic ecological and chemical evolution the climate system. Develop, deliver, and support the Community Climate System Model (CCSM).

Table 1 (continued)

Science domain	Example science driver
Combustion	Develop cleaner-burning, more efficient devices for combustion.
Engineering	Develop and correlate/validate large-scale computational tools for flight vehicles. Demonstrate the applicability and predictive accuracy of computational fluid dynamics (CFD) tools in a production environment. Investigate flight vehicle phenomena such as fluid-structure/flutter interaction, and control surface free-plays.
Fusion	Resolve fundamental science and engineering questions in fusion reactor technology. Understand interactions that both RF wave and particle sources have on extended MHD phenomena. Understand and control plasma turbulent fluctuations that can cause loss of heat needed to maintain the fusion reaction.
High energy physics	Seek to find the Higgs particles thought to be responsible for mass, using the Large Hadron Collider (LHC) physics program. Seek to find evidence of supersymmetry (SUSY), a necessary element of String Theory that may unify all of nature's fundamental interactions.
Materials science	Understand the initiation of failure in a local region, the appearance of a macro-crack due to the coalescence of subscale cracks, the localization of deformation due to coalescence of voids, the dynamic propagation of cracks or shear bands, and all causes leading to eventual fragmentation and failure of a solid.
Nanoscience	Understand the quantitative differences in the transition temperatures of high temperature superconductors. Understand and improve colossally magneto-resistive oxides and magnetic semiconductors. Develop new switching mechanism in magnetic nanoparticles for ultra high density storage. Simulate and design molecular-scale electronics devices. Elucidate the physical-chemical factors mechanisms that control damage to DNA.
Nuclear energy	Design and deploy efficient and safe closed nuclear fuel cycle facilities, including next-generation power generation and recycle reactors, separations reprocessing facilities, and fuel fabrication/storage facilities.
Nuclear physics	Develop ways to describe nuclei whose properties cannot be measured (e.g., thermal nuclear properties in the mass 80–150 region).

Leadership Computing (LC) will make possible many breakthroughs in science in the next decade. Many exciting opportunities present themselves as evidenced by accomplishments already attained in science disciplines supported by petascale computing. These opportunities are realizable, but they are also confronted with challenges, uncertainties, and issues on the horizon.

For a science application to be mission relevant, alignment with the DOE ASCR Strategic Plan is important, which includes (1) enabling new materials through nanoscience; (2) enabling the design and engineering of fusion power plants to produce energy without CO_2; (3) understanding the regional effects of global climate change; (4) developing new bacteria that can produce hydrogen, sequester carbon, and clean up toxic wastes; (5) understanding the fundamental nature of matter; and (6) understanding the processes that underpin combustion of fossil fuels to reduce pollution and increase efficiency. Business

requirements, which broadly address science motivation and impact, are elicited from all LCF projects by probing the

- need for LC and mission relevancy;
- science questions (what/when/why) being answered with LC;
- need for and quality of simulation validation;
- identification of the clients, customers, and users, defined as those who pay for product development, pay for product, and use the product; and
- list of all end products.

Appendix A of this document is a glossary of application codes; Appendix B provides information on project allocations and usage; Appendix C gives the charter and mission of the Applications Requirements Council; Appendix D gives the code project questionnaire developed by the Advanced Scientific Computing Advisory Committee (ASCAC); and Appendix E is the survey of acceptance and early access science applications.

References

1. Kenneth J. Roche et al., *Application Software Case Studies in FY05 for the Mathematical, Information, and Computational Sciences Office of the U.S. Department of Energy,* Internal document on file at the Office of Advanced Scientific Computing Research, DOE point of contact: Dr. Michael Strayer, DOE Office SC-21; ORNL point of contact: Kenneth J. Roche, rochekj@ornl.gov, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6173, 2006.
2. Wibe A. de Jong and Theresa L. Windus, Editors, *Scientific Challenges: Linking Across Scales,* Pacific Northwest National Laboratory, PNNL-15144, July 2005.
3. S. C. Jardin, Editor, *DOE Greenbook: Needs and Directions in High Performance Computing for the Office of Science,* Princeton Plasma Physics Laboratory, PPPL-4090, June 2005.
4. U.S. Department of Energy Office of Science, *Scientific Discovery Through Advanced Computing: Progress and Opportunities,* D. Gracio, J. Mitchell, N Samatova, T. Straatsma, Eds., Point of contact: Deborah Gracio, debbie.gracio@pnl.gov, Draft, September 2005.
5. U.S. Department of Energy, Office of Science, *A Science-Based Case for Large-Scale Simulation,* Vol. 1, July 2003 (see http://www.pnl.gov/scales/).
6. U.S. Department of Energy Office of Science, *A Science-Based Case for Large-Scale Simulation*, Vol. 2, September 2004 (see http://www.pnl.gov/scales/).
7. High-End Computing Revitalization Task Force, *Federal Plan for High-End Computing,* May 2004.

SCIENCE QUALITY AND PRODUCTIVITY REQUIREMENTS

Science quality and productivity requirements define simulation capabilities and output that LCF systems must support so that science applications can embody more predictive models and run to completion more quickly so that more accelerated decision-making, discovery, and understanding results.

From one point of view, science quality and productivity requirements directly reflect the DOE Office of Science "Joule Metrics," namely, efficiency and improvement in simulation time to solution for a given problem size (one measure of productivity) and constant (or improving) simulation time to solution for ever-larger problems (one measure of quality) (Table 2). In a broader sense, science quality and

Table 2. Science investigations and achievements possible on a 1-PF LC system for specific application codes in relevant science domains

Science domain	Science achievements possible
Accelerator physics	Design the ILC.
Astrophysics	Determine the explosion mechanisms of core-collapse and Type Ia supernovae.
Biology	Help to make biofuels economically feasible.
Chemistry	Improved processes for clean coal, hydrogen production and storage, and catalyst design.
Climate	Simulate the dynamic ecological and chemical evolution the climate system.
Combustion	Develop a fundamental understanding of high-efficiency, low-emissions combustion devices required for transportation and power generation.
Engineering	Demonstrate the applicability and predictive accuracy of continuum engineering tools in a production environment.
Fusion	Improve our capability for predicting and optimizing the performance of burning plasmas. Attain a more realistic assessment of ignition margins using more accurate calculations of steady-state temperature and density profiles for ions, electrons, and helium ash.
High energy physics	Close in on the unifying theory for all of nature's fundamental interactions.
Materials science	Predictive simulation of brittle and ductile materials subjected to high-speed loads.
Nanoscience	Hone in on the theory for high temperature superconductors; design and construct nanoparticles for specific tasks (magnetic storage, electronic devices, semiconductors, etc.)
Nuclear energy	A virtual "flight simulator" for an operating closed fuel cycle facility.
Nuclear physics	Accurate nuclear properties for material whose properties cannot be measured.

productivity requirements define properties that LC systems must possess so that science applications are able to embody higher-fidelity physics models (becoming more predictable) and to run to completion quicker so that researchers can more quickly arrive at decisions, discovery, or understanding (allowing the researcher to be more productive) (Table 3). Science quality and productivity requirements are elicited from all LCF projects by understanding the

- quality (fidelity of physics models) of current applications and how this might improve with LC system specifications [e.g., peak speed from 25 teraflops (TF) to a sustained 1 petaflops (PF)];
- productivity of science output, and how this might change with LC system specifications (e.g., peak speed from 25 TF to a sustained 1 PF);
- work flow of the science simulations being performed (e.g., simulation turn-around times, problem setup times, use cases, bottlenecks);
- extent to which applications are validated (physical models compared against experimental data) and the breadth/depth/quality of simulation testing needed to improve the validation state;
- confidence level (level of predictability) of current applications, whether or not this be quantified (e.g., error bars), and how it might change as a function of LC system specifications.

Table 3. Increase in science simulation fidelity possible with a 1-PF LC system for specific application codes in various science domains

Science domain	Code	Fidelity at 25 TF	Fidelity at >1 PF
Astrophysics	Chimera	3-D hydro simulations to follow the shock evolution out to several times the stalled shock radius.	Improved transport scheme (Boltzmann S_n) Improved nuclear kinetics (150 species versus an alpha network)
Astrophysics	Vulcan2D	2-D multigroup, time-dependent radiation hydrodynamics with 10,000-km and 2-s resolution.	3-D multigroup, time-dependent radiation hydrodynamics. More integration time, more state variables, increased complexity reaction networks.
Biology	LAMMPS	Dynamics of 700K-atom systems for 5–10 ns of model time per day of simulation time.	Multimillion atom systems evolved for 0.1–1.0 ms.
Climate	CCSM	Eulerian spectral atmospheric circulation model with diurnal cycle resolved columnar radiation and moist convection (CAM3), Brian-Cox free-surface ocean model with Gent-McWilliams eddy parameterization (POP1.4), dynamic sea-ice model with visco-plastic rheolgy (CICE), land surface model with soil, river and vegetation components (CLM3).	Tropospheric chemistry (100 species), dynamic vegetation, terrestrial carbon pools, ocean ecosystems, land ice sheets, stratospheric chemistry, full sulfur cycle, increase in ensemble size for climate change studies, coupled-ocean eddy-resolving simulations, cloud microphysics, realistic land-use patterns, tropical event simulation on climate timescales.

Table 3 (continued)

Science domain	Code	Fidelity at 25 TF	Fidelity at >1 PF
Climate	MITgcm	4-km horizontal and variable (30 levels) vertical resolution in a 2000 by 4000 km domain 2000 m deep. Time step of 500 s and integration of the primitive equations (3 diagnostic and 2 prognostic variables) for several decades (20–40 years).	Understanding the role of non-hydrostatic physics and internal wave breaking in deep ocean mixing, required to close the heat and salinity budget suggested by the current sinking rates in high latitudes and in the establishment of the thermohaline circulation.
Combustion	S3D	Chemical mechanism for CO/H_2 and reduced mechanism for CH_4 and molecular transport model. 2.5 decades of time and length scales resolved for reactive turbulent flow. Moderate Reynolds numbers of 5–15K.	Increase Reynolds numbers to >20K, (consistent with internal combustion engines) and pressures to 10–20 atm. Chemical mechanisms include multi-stage n-heptane ignition.
Fusion	GTC	Gyrokinetic ions with drift-kinetic electrons and electrostatic perturbations. Resolved time scale is the electron transit time and the resolved length scale is the ion gyroradius.	Integrated simulation with gyrating ions and drift-kinetic electrons with electromagnetic perturbations by resolving ion cyclotron waves and electron skin depth. Transport time scale simulations (evolving plasma background equilibrium).
Fusion	GYRO	Test of first-principles models of plasma turbulence against measured levels of heat and particle transport in tokamaks (DIII-D, C-mod, NSTX) to build reduced models of plasma transport to predict performance of prototypes.	ITER performance predictions. Possible to introduce feedback loop to adjust input profiles to match target heat and particle flows for truly predictive simulation.

Model and Algorithm Requirements

Application models represent functional ("what") requirements that drive the need for certain numerical algorithms and software implementations. They are also often pre-determined by the given features and specifications in LC systems (Table 4). A choice and specification of LCF system attributes (e.g., peak speed or node memory capacity) tends to constrain the functional attributes employed usefully in a given physical model on that system. For example, attributes such as the following all depend in part upon the LCF system for which implementation of the models was targeted.

- model state variables (how many now, how many planned in the future),
- model characteristics [partial differential equations (PDE) or ordinary differential equation (ODE); deterministic or stochastic: formulation of equations],
- the presence of multiple, simultaneous phenomena, and the required degree of coupling,
- the domain of dependence (local with specific patterns, global), and
- data dependency (degree of parallelizability)

Table 4. Examples of how physical model attributes might change on a 1-PF LC system for specific application codes in various science domains

Science domain	Code	Current physical model attributes	Physical model attributes at >1 PF
Astrophysics	Chimera	Deterministic nonlinear integro-PDEs; 63 variables.	High-resolution energy and angle phase space and a 200-species nuclear network; >1000 variables.
Climate	CCSM	Deterministic nonlinear PDEs; 5–10 prognostics and ~100 diagnostic variables.	Could add another ~100 diagnostic variables for biogeochemical processes.
Climate	MITgcm	Deterministic nonlinear PDEs; 3 prognostic and 2 diagnostic variables.	Could add stochastic component. 5 prognostic and 1 diagnostic variables; can vary key forcing parameters to study the response to changed climate scenarios.
Combustion	S3D	Deterministic nonlinear PDEs; 16 variables.	Better chemical kinetics could result in 75 variables.
Fusion	GTC	Vlasov equation in Lagrangian coordinates as ODEs, Maxwell equations in Eulerian coordinates as PDEs, and collisions as stochastic Monte Carlo processes; 2 field equations and 5 phase variables per particle.	5 field equations and 6 phase variables per particle.
Fusion	GYRO	2 field, no feedback.	3 field with profile feedback.

After a physical model has been postulated, the application developer must devise and/or use one or more algorithms to generate numerical solutions to the model as formulated. For most of the applications surveyed, the physical models tend to be sets of coupled linear and nonlinear PDEs and ODEs. Most of the time application model requirements inherently imply algorithm requirements because they are closely tied to the algorithms chosen to find numerical solutions.

Application algorithm requirements are design ("how") requirements that clarify the ramifications of these choices in the science applications on LCF specifications. Algorithm requirements are elicited from each application by understanding:

- the parallelism paradigm (distributed, domain replicated, coupled distributed models, etc.) and method for implementation [message-passing interface (MPI) tasks, threads, etc.];
- scalability (how many execution threads/tasks can be handled) and any identified obstacles to scalability;
- the extent of algorithm convergence, accuracy, and verification;
- solution methods (linear/nonlinear sets of equations, matrix properties, etc.), categorized according to Colella's "Seven Dwarfs" *(Table 5); and
- algorithm adaptivity as a function of space, time, and data.

* *Defining Software Requirements for Scientific Computing,* Phillip Colella (2004).

Table 5. The "seven dwarfs" categorization of algorithms employed by specific application codes in various science domains

Science domain	Code	Structured grids	Unstructured grids	FFT	Dense linear algebra	Sparse linear algebra	Particles	Monte Carlo
Accelerator physics	T3P		X			X		
Astrophysics	CHIMERA	X			X	X	X	
	VULCAN/2D		X		X			
Biology	LAMMPS			X			X	
Chemistry	MADNESS		X		X			
	NWChem			X	X			
Climate	CAM	X		X			X	
	POP/CICE	X				X	X	
	MITgcm	X				X	X	
Combustion	S3D	X						
Fusion	AORSA	X		X	X			
	GTC	X				X	X	X
	GYRO	X		X	X	X		
Geophysics	PFLOTRAN	X	X			X		
Materials science	QMC/DCA				X			X
	QBOX			X	X		X	
Nanoscience	CASINO						X	X
	LSMS	X			X			
Nuclear energy	NEWTRNX		X		X	X		
Nuclear physics	NUCCOR				X			
QCD	MILC	X						X

Note: The "X" denotes a particular algorithm is utilized by that code.

For each one of these areas, it is important to understand how and where these algorithms are likely to change in the next 5 years or when a 1-PF system becomes available.

Several trends are noteworthy in the "seven dwarfs" categorization of codes in

- The seven algorithm types are scattered broadly among science domains, with no one particular algorithm being ubiquitous and no one algorithm going unused.
- Structured grids and dense linear algebra algorithms are the most widely used algorithms (used by over half of the representative codes), hence system attributes such as node peak flops and memory capacity, memory latency, and interconnect latency will be important (see the section on Runtime Requirements in this document).

- Particle-based and Monte Carlo algorithms, which have similar properties from a system standpoint, are also broadly used, and can tax interconnect latency and in some cases node memory capacity, depending upon implementation and usage.

SOFTWARE REQUIREMENTS

Applications impose software requirements on LC systems in the form of programming models and languages and I/O and math libraries. This well-defined set will become more complex with the advent of multicore and accelerator-based architectures.

Application software requirements are design ("how") requirements that elucidate the ramifications of current and planned science application software implementations on LC systems (Table 6). These requirements not only lead in part to a predefined set of tools that must be present on LC systems, but they also help to point out potential pitfalls and dead ends in some current software choices. Science application software requirements are elicited from each application by analyzing the following:

- the programming languages, external libraries, and tools used and needed and where any productivity bottlenecks might exist;
- the breadth, depth, and quality of software verification and testing employed;
- the software engineering attributes (best practices, team development); and
- the quality and maturity of software (as judged by the application team).

Table 6. Functional software requirements (and options) for specific application codes in various science domains

Science domain	Code	Programming language	Programming model	I/O libraries	Math libraries
Accelerator design	T3P	C/C++	MPI	NetCDF	MUMPS, ParMETIS, Zoltan
Astrophysics	CHIMERA	F90	MPI	HDF5 (pNetCDF)	LAPACK
	VULCAN/2D	F90	MPI	HDF5	PETSc
Biology	LAMMPS	C/C++	MPI		FFTW
Chemistry	MADNESS	F90	MPI		BLAS
	NWChem	F77, C/C++	MPI, Global Arrays, ARMCI		BLAS, ScaLAPACK, FFTPACK
Climate	CAM	F90, C (CAF)	MPI (OpenMP)	NetCDF	(SciLib)
	POP/CICE	F90 (CAF)	MPI (OpenMP)	NetCDF	
	MITgcm	F90, C	MPI (OpenMP)	NetCDF	
Combustion	S3D	F90	MPI		

Table 6 (continued)

Science domain	Code	Programming language	Programming model	I/O libraries	Math libraries
Fusion	AORSA	F77, F90	MPI	NetCDF	ScaLAPACK, FFTPACK
	GTC	F90, C/C++	MPI (OpenMP)	MPI-IO, HDF5, NetCDF, XML	PetSC
	GYRO	F90, Python	MPI	MPI-IO, NetCDF	BLAS, LAPACK, UMFPACK, MUMPS, FFTW (SciLib, ESSL)
Geophysics	PFLOTRAN	F90	MPI		BLAS, PetSC
Materials science	LSMS	F77, F90, C/C++	MPI2	HDF5, XML	BLAS, LAPACK
	QBOX	C/C++	MPI	XML	LAPACK, ScaLAPACK, FFTW
	QMC	F90	MPI		BLAS, LAPACK, SPRNG
Nanoscience	CASINO	F90	MPI		BLAS
	VASP	F90	MPI		BLAS, ScaLAPACK
Nuclear energy	NEWTRNX	F90, C/C++, Python		HDF5	LAPACK, PARPACK
Nuclear physics	NUCCOR	F90	MPI	MPI-IO	BLAS
QCD	MILC, Chroma	C/C++	MPI		

For each one of these areas, it is important to understand how and where software needs to change and is likely to do so in the coming years (Table 7). A very important theme emerged upon analyzing application software requirements: too many application software designs are monolithic and not component-based. Such designs will not be able to adequately exploit future LC architectures, where the ability to use optimized, hardware-specific middleware (e.g., math libraries) will be critical.

System software is considered here to be the software associated with the operating system, file system, and run-time libraries.

Table 8 presents proposed solutions to specific software requirements.

Table 7. Typical features and associated suggested requirements for components of an LC system software stack

System software feature	Requirement
Mathematical libraries	BLAS, LAPACK, SCALAPACK, PETSc, SuperLU, and Parallel FFT tuned to the LC systems and modified to exploit multicore.
Communication library	High-performance, fault-tolerant communication library able to deal with dead nodes.
Specialized mathematical libraries	Specialized, high-performance O(N) libraries (USFFT, KFFMM, MRA, LSR, Generalize Gaussian Quadrature) optimized for the LC systems.
Lightweight OS kernel	Scalable and robust kernel with support for multicore processors as an SMP node.
I/O and storage	Increased scalability and updated algorithms for data and metadata servers.
Reliability and fault tolerance	Development of advanced systems software enabling applications to have and use built-in fault handling.
Advanced debugging	Comparative debugging tools to support the simultaneous execution of two versions of an application, allowing the selection of comparison points for verification.
Automatic performance analysis	Easy-to-use, automated performance tools able to handle large amounts of data. Development of an infrastructure to support scalability and automation.
Integrated compilation	Compilation environment for applications simultaneously targeted for different systems (scalar/vector processors, FPGAs, stream-based coprocessors, etc.).

Table 8. Proposed solutions to specific requirements for components of the NCCS LCF software stack

Requirement	NCCS LCF software stack
Resource manager/scheduler	Torque, Moab, CRMS (will become ALPS)
Workflow tools	Kepler, bbcp
User mgmt, ticket system, accounting	ORNL Resource Accounting and Tracking (RATS), RT
Security and fault detection	Nagios, Inmon, OSIRIS, SNORT/BRO
Compilers	PGI, Pathscale
Vendor math libraries	SciLIB, ACML
Community math libraries	FFTW, PETSc, LAPACK, ScaLAPACK, Atlas, Goto BLAS
Programming languages	Fortran, C/C++, CAF
Performance and debugging tools	CrayPat, Apprentice, TotalView, PAPI
Parallel I/O libraries	HDF5, pNetCDF, MPI-IO
MPI	MPT
Low-level communication layers	Portals, ARMCI
Shared memory layers	OpenMP, Threads
CN and ION kernels, CIOD	CVN, CNOS (Linux) and SUSE
Visualization and data analysis	VisIt, EnSight, IDL, AVS/Express, Parallel R, VTK, Matlab
Production file system	Lustre
Archive tools	hsi, htar

RUNTIME REQUIREMENTS

Detailed knowledge of the application runtime footprint, expressed in a relative sense, allows reliable extrapolation to future LC systems.

Application runtime requirements are design ("how") requirements that specify the application "footprint" on LC systems. The footprint is what the system sees from the application while it is executing: its memory usage, memory patterns, communication usage and attributes, I/O usage and attributes, etc. Often applications requirements are solely determined by collecting runtime attributes only. While these are important, they should be derivable from detailed knowledge of other types of requirements (model, algorithm, software), and they do not help to elucidate future applications needs unless they are specified in a normalized sense. An example of a useful requirement is a statement that an application must dump 20% of its simulation images every 100 time steps. A less useful requirement is a statement that an application needs 100 TB of locally attached disk space. If runtime requirements are expressed in a relative or normalized sense, then more accurate and reliable extrapolations to future systems are possible. Application runtime requirements are elicited by probing its

- I/O model and volume (size, bandwidth, parallel scalability),
- extent of indirect addressing,
- ability to execute on a heterogeneous system,
- need for and ability to do dynamic data repartitioning,
- extent of load imbalance,
- communication patterns (global, local, size of message, number of messages),
- performance bottlenecks and metrics, and
- memory usage.

A given LCF system has many attributes that uniquely characterize it relative to other systems, but the following 12 attributes in particular have been found to be useful and important to consider from the application's perspective:

- peak flops per node,
- mean time to interrupt (MTTI),
- wide area network (WAN) bandwidth,
- node memory capacity,
- local storage capacity,
- archival storage capacity,

- memory latency,
- interconnect latency,
- disk latency,
- interconnect bandwidth,
- memory bandwidth, and
- disk bandwidth.

For each of these 12 system attributes, certain behaviors and properties of a given application may expose one particular attribute relative to another. Table 9 summarizes application behaviors and properties that serve as drivers for system attributes.

Table 9. Science application behavioral and algorithmic drivers for LC system attributes

LC system attribute	Application algorithms driving a need for this attribute	Application behaviors driving a need for this attribute
Node peak flops	Dense linear algebra, FFT, sparse linear algebra, Monte Carlo	Scalable and required spatial resolution low; would benefit from a doubling of clock speed; only a problem domain that has strong scaling, completely unscalable algorithms; embarrassingly parallel algorithms.
Mean time to interrupt	Particles, Monte Carlo	Naïve restart capability; large restart files; large restart R/W time.
WAN bandwidth	Long time evolution, multiphysics, multiscale	Community data/repositories; remote visualization and analysis; data analysis.
Node memory capacity	Dense linear algebra, sparse linear algebra, unstructured grids, particles	High DOFs per node, multi-component/multi-physics, volume visualization, data replication parallelism, restarted Krylov subspace with large bases, subgrid models.
Local storage capacity	Particles, out-of-core algorithms	High-frequency/large dumps, out-of-core state, debugging at scale.
Archival storage capacity	Long time evolution, multiphysics, multiscale	Large data (relative to local storage) that must be preserved for future analysis, for comparison, for community data (e.g., EOS tables, wind surface, and ozone data); expensive to recreate; nowhere else to store.
Memory latency	Sparse linear algebra, unstructured grids	Data structures with stride-one access patterns (e.g., cache-aware algorithms); random data-access patterns for small data.
Interconnect latency	Structured grids, particles, FFT, sparse linear algebra (global), Monte Carlo	Global reduction of scalars; explicit algorithms using nearest-neighbor or systolic communication; interactive visualization; iterative solvers; pipelined algorithms.
Disk latency	Out-of-core algorithms	Naïve out-of-core memory usage; many small I/O files; small record direct-access files.
Interconnect bandwidth	FFT and other spectral methods, coupled models	Large messages, global reductions of large data; implicit algorithms with large DOFs per grid point.

Table 9 (continued)

LC system attribute	Application algorithms driving a need for this attribute	Application behaviors driving a need for this attribute
Memory bandwidth	Sparse linear algebra, unstructured grids	Large multidimensional data structures and indirect addressing; lots of data copying; lots of library calls, requiring data copies if algorithms require data retransformations; sparse matrix operations.
Disk bandwidth	Out-of-core algorithms	Reads/writes large amounts of data at a relatively low frequency; read/writes large amounts of large intermediate temporary data; well-structured out-of-core memory usage.

A qualitative prioritization of these system attributes for each domain science is shown in Table 10. Priorities are presented according to color: green is highest, yellow is moderate, and grey is lowest priority. A high priority (green) attribute should be maximized over a lower (yellow or grey) priority attribute for a computer system designed for that science domain. For each domain, four attributes were given a high (green) priority, four a moderate (yellow) priority, and four a low (grey) priority.

Table 10. Three-tier prioritization of 12 system attributes for relevant science domains*

System attribute	Climate	Astrophysics	Fusion	Chemistry	Combustion	Accelerator physics	Biology	Materials science
Node peak flops								
MTTI								
WAN network bandwidth								
Node memory capacity								
Local storage capacity								
Archival storage capacity								
Memory latency								
Interconnect latency								
Disk latency								
Interconnect bandwidth								
Memory bandwidth								
Disk bandwidth								

*In each science domain, green denotes an attribute with the highest priority for maximizing, yellow is moderate priority, and grey lowest priority.

An example of moving from qualitative to more quantitative runtime requirements is the analysis of what currently constitutes a single simulation for selected application codes. Such analysis helps to validate the importance of system attributes for these codes. Table 11 presents typical development

characteristics and runtime requirements of a single simulation for selected application codes on the NCCS LCF systems.

Table 11. Typical development characteristics and runtime requirements of a single simulation (job) for selected application codes on the NCCS LCF systems circa June 2006

Science domain	Code	Code attributes	Job size (nodes, time)	Storage capacity needs (local, archive)	Node memory capacity needs	Number of queue dwell times needed for full simulation
Accelerator design	Omega3D	9 years old, 173-K C++ LOC, 12 developers	128–256, 24 hours	1 TB, 12 TB	8 GB	3–4
Astrophysics	CHIMERA	Components 10–15 years old, 5 developers, F90	128–256, 24 hours	300 GB, 2 TB	≥2 GB	10–15
Astrophysics	Vulcan2D	9 developers, F90	48, 24 hours	300 GB, 5 TB	2 GB	30
Climate	CCSM	Components 20 years old, 690 K Fortran LOC, over 40 developers	250, 24 hours	5 TB, 10 TB	2 GB	10–30
Combustion	S3D	16 years old, 100 K Fortran LOC, 5 developers	4000, 24 hours	10–20 TB, 300 TB	1 GB	7–10
Fusion	GTC	7 years old, ~30 developers	4800, 24 hours	10 TB, 10 TB	2 GB	4–5
Nuclear physics	NUCCOR	3 years old, 10 developers, F90	200–1000, 4–8 hours	300 GB, 1 TB	2 GB	1

DATA ANALYSIS AND DATA MANAGEMENT REQUIREMENTS

Application data analysis and management requirements are more quantifiable, dictating onerous constraints on LC system I/O infrastructure unless end-to-end workflows change.

Data analysis and management requirements specify end-to-end application analysis characteristics and needs on LCF systems. These requirements are collected for each application by determining:

- the analysis tools (visualization, data mining, etc.) needed,
- file system storage needs per simulation,
- the maximum desired read/write time as a percentage of simulation time; and
- archival storage needs per simulation.

Given the above, several other important application-specific data analysis requirements directly follow; for example, output bandwidth (GB/s) from the LC system to local storage. The most demanding data management requirements often come from applications that incorporate multiphysics and multiscale models. This kind of coupling leads to high dimensionality in evolved quantities (e.g., radiation fields, chemical and nuclear species, and particle phase spaces). These applications also tend to involve long time evolutions. Therefore, large multidimensional datasets are output at many regular intervals to allow for analysis of time-dependent correlations and the overall evolution of the modeled systems. The particular science objectives for these types of applications are often directly relatable to a set of resolution requirements—in time, space, and tangent or phase space—which in turn determines the overall size of the datasets output and their number. This scaling also holds for the I/O requirements for checkpoint (restarts). The infrastructure requirements for the LCF that stem from these application requirements ultimately set the scale of the scientific simulations that can be performed on the system.

The I/O requirements for LCF systems can be broken down into two distinct categories, namely, those required to

- output results in the form of restart dumps and other analysis files, and
- postprocess data files for analysis and visualization.

The "output" portion of I/O requirements can be determined for the LCF by examining the needs of the largest data-producing codes on the current systems. The current largest data producers are the following application codes: CHIMERA, GTC, S3D, T3P/Omega3P, and POP. I/O requirements were therefore explicitly elicited from users and developers of these codes, with the results highlighted in Table 12 for per-simulation requirements on a 1-PF LC system with 200 TB of memory. Most application

Table 12. Typical per-simulation I/O requirements for the largest data-producing application codes on the NCCS LCF systems*

Science domain	Code	Restart file size	Restart frequency	Analysis file size	Analysis dump frequency	File type	Analysis tools
Astrophysics	CHIMERA	160 TB	1/hour	160 TB	1/hour	pnetCDF or binary, collective	IDL, xmgrace, EnSight
	VULCAN/2D	20 GB	1/day	200 GB	10/day	Binary, HDF5	VTK, Open-DX, IDL
Climate	POP	26 GB	1/hour	1.4 GB per field	1/minute	Binary, 1 serial file	IDL, NCAR graphics
Combustion	S3D	5 TB	1/hour	5 TB	2/hour	Binary, individual files	TecPlot, VisIt, Post_S3D, Matlab
Fusion	GTC	20 TB	1/hour	10 GB	1/minute	Binary, individual files	IDL, gnuplot, Matlab, AVS/Express, EnSight
Fusion	GYRO	50 GB	1/hour	10 GB	1/minute	Binary, collective	IDL, VTK, Asymptote

*A 1-PF LC system with 200 TB of memory is the assumed system.

codes were found to write restart files on a per-processor basis to get the best performance on the system. Ideally, users would like to write out the data via pNetCDF or parallel HDF5, thereby producing a single inode per restart dump. Users also require that the system have minimal impact while writing the restart and analysis files, namely by keeping I/O overhead at less than 5% of the total run time. This study has found that the users would ideally like to generate restart files ranging from 10 to 80% of the total memory on the nodes used in the run, but often do much less (like 1–20%) because the I/O overhead would be much larger than 5%. This information, along with a conservative estimate of MTTI, helps set the restart dump frequency, which in turn can be used to determine a minimum write bandwidth to local storage. The prescription is given in Table 13; required local storage bandwidth can be reasonably estimated as the ratio of restart file size to time tolerated by the user necessary to write out the data. The time tolerated for output is usually some small fraction (5–10%) of the restart output periods. Application restart output periods are usually 1–2 hours for LC systems, set ultimately by the system MTTI or queue dwell-time maximum, which is often 24 hours (or less).

Table 13. Prescription for estimating local storage bandwidth requirements for science applications on LC systems*

Variable	Description	Typical values
M	Total system memory	100–400 TB for a 1-PF system
f	Fraction of application runtime memory captured and written out per restart	20–80%
T	Runtime intervals between successive restart file outputs	1–3 hours when MTTI is 12–24 hours or maximum queue runtimes ~24 hours
O	Maximum allowable fraction of total runtime devoted to I/O operations	10%
B	Required bandwidth to local storage	$= \frac{fM}{TO}$

*I/O activity is assumed to follow a periodic saw tooth pattern, namely short bursts of I/O activity followed by longer periods of I/O inactivity.

Using the range of local storage bandwidth requirements, users can compute bandwidth requirements as shown in Table 14. As an example, assume an application is running on the entire 1-PF system, occupying at or near the entire 200-TB system memory, and must write out every hour a restart file whose size is roughly 10% of the occupied memory, or 20 TB. If the tolerable I/O overhead is 5%, then the 20-TB file must be written out in 3 minutes (180 seconds), corresponding to an output bandwidth of 111 GB/s. This analysis can be performed for all ranges spanning known requirements.

Table 14. Estimates of science application local storage bandwidth requirements using the prescription outlined in Table 13*

Restart file size/ total system memory	Restart period (hours)	Allowable I/O overhead (%)	Required local storage bandwidth (GB/s)
0.10	1	5 10	111 56
	2	5 10	56 28
0.20	1	5 10	222 111
	2	5 10	111 56
0.80	1	5 10	888 444
	2	5 10	444 222

*Simulations are assumed to run on a full 200-TB LC system.

It is important to note in this analysis that use of asynchronous I/O by the application was not assumed, either for historical reasons or because memory constraints render buffering I/O unfeasible for the largest simulations. Nevertheless, the use of asynchronous I/O could markedly reduce bandwidth requirements, introducing at the same time a new set of software infrastructure requirements. Bandwidth requirements could easily be reduced by an order of magnitude (relative to Table 14) if applications employed asynchronous I/O; this will be investigated further.

Storage requirements for all produced data can be estimated in a number of ways. A good estimate can be determined by simply aggregating typical restart and analysis file sizes for each application and multiplying by the total number of projects according to the prescription laid out in Table 15 and Table 16. Given the total system memory, the total number of LC projects, a reasonable estimate of project output file size per simulation, and an average number of simulations whose output will be retained on local storage, a local storage capacity estimate follows from their collective product.

Table 15. Prescription for estimating local storage capacity requirements for science applications on LC systems

Variable	Description	Typical values
M	Total system memory	100–400 TB for a 1-PF system
P	Total number of projects with LC allocations annually	20–40
F	Fraction of application runtime memory captured and written out per restart	20–80%
R	Average number of simulations per project whose output is retained on local storage	10–20
C	Required local storage capacity	$= fMPR$

If, for example, each of 20 projects required saving 20% of the total application memory per simulation (with applications occupying the entire system), and 10 such simulations (~1 per month) were retained, then 8 PB of local storage capacity is required to support this workload.

Archival storage requirements are more difficult to estimate because the use cases for archived data vary widely from application to application. A reasonable methodology is to scale current archival storage usage with a system attribute that is directly tied to data generation (e.g., peak flops, total memory, memory bandwidth). From an application point of view, however, application storage requirements are mostly driven by total memory used per simulation, hence, total system memory scaling is most

Table 16. Estimates of science application local storage capacity requirements using the prescription outlined in Table 15

Number of projects	Restart file size/total system memory	Number of runs per project retained on local storage	Required local storage capacity (PB)
10	0.20	2 10	0.8 4.0
	0.80	2 10	3.2 16.0
20	0.20	2 10	1.6 8.0
	0.80	2 10	6.4 32.0
40	0.20	2 10	3.2 16.0
	0.80	2 10	12.8 64.0

*An LC system memory of 200 TB is assumed.

appropriate. It is interesting, however (as shown in), that estimates of archival storage capacity needs based on any of these three system attributes yield results for the 1-PF system that are all within a factor of two of one another. The memory-scaled or memory bandwidth-scaled estimates are likely to be the most reliable, however, because application output requirements directly follow from memory used per application simulation. Another defensible approach for estimating archival storage requirements is to take the local storage requirements (e.g., those in Table 16) and assume that some or all of these data must also be archived (admittedly neglecting any experimental data storage requirements). If, for example, the requirements in Table 16 are expected to occur annually during those years the 1-PF system is deployed, then archival storage requirements for any given year would represent the

Table 17. Estimates of science application archival storage capacity requirements based on scaling current capacities with either system memory, memory bandwidth, or peak flops

System attribute assumed to govern archival storage requirements	Estimated capacity needs by end of CY06 (PB)	Estimated 250-TF capacity needs (PB)	Estimated 1-PF capacity needs (PB)
Memory	2.8	4.6	15.9
Memory bandwidth	3.8	10.8	36.0
Peak flop rate	3.6	7.1	18.5

accumulation of local storage requirements over some previous number of years. Other system characteristics can lead to considerably different results, as archival storage requirements are cumulative over the lifetime of the system.

SUMMARY AND RECOMMENDATIONS

Leadership Class Facilities must engage in a regular and evolving applications requirements process that is rigorous and quantitative. This process is difficult, time-consuming, yet necessary. High-consequence decisions about current and future systems informed by this process are virtually guaranteed to deliver what is best for accelerating scientific discovery and understanding.

Establishing a formal, rigorous, and useful requirements management process is very challenging when applied to breakthrough science applications for leadership computing, where the research is by its very nature exploratory and high risk. The requirements process must always evolve, continuing to improve as guided by lessons learned, just as this document must be a living document, ever-changing to keep up with the applications themselves. Computational science requirements for LC computing flow both ways—LCF systems set requirements for the science applications just as the science applications must set requirements for the LCF systems. Nevertheless, given current experience, we provide the following set of preliminary specific observations and recommendations:

SCIENCE

- The annual number of allocated projects must be small (<25) to ensure science output and access to LCF resources are maximized.
- Science roadmaps in every field call for not only increased fidelity but also increased productivity upon access to a petascale LCF system. An example of increased fidelity is executing a larger simulation in the same turn-around time as on a smaller system; an example of increased productivity is executing the same size simulation in a shorter turn-around time.

MODELS AND ALGORITHMS

- As applications are ported to, developed on, and used on petascale LCF systems, the change in physical models employed is likely to be more evolutionary than revolutionary. The prototypical example is the solution of nonlinear PDEs—a petascale LCF system affords more spatial and temporal resolution, which modern solution methods should easily support given a correct formulation. A drastic change in physical models (e.g., from a deterministic PDE to a nondeterministic model) as motivated by access to a petascale LCF system is not likely to be the norm. Exceptions could be climate, biology, and chemistry, among others.

- Parallel algorithm maturity and efficiency vary widely from one field to another and from one code to another. For example, fields focused on "atomistics" (nanoscience, materials science, chemistry, biology, etc.) have parallelism challenges that are unique enough to make it difficult for other fields to contribute useful approaches.
- A "seven dwarfs" algorithm analysis of applications indicates, not surprisingly, that there are no algorithm "sweet spots," thereby disallowing an LCF system to pursue a hardware architecture designed to specifically optimize a particular algorithm (e.g., FFT).

Software

- Standard programming languages (e.g., Fortran, C, and C++) remain the scientific computing staple on LCF systems. To a lesser extent, Co-array Fortran and scripting languages like Python are also needed, but a demand for brand new and/or unanticipated languages is not evident at present.
- Parallel programming strategies continue to emphasize MPI, along with, in some cases, OpenMP and Global Arrays. Other paradigms need to be examined, at least in prototype form, in order to demonstrate proof of principle.
- "Critical" math libraries needed by a large fraction of applications include BLAS, LAPACK, FFTPACK, FFTW, and PETSc. Others needed (but not as prevalent) include ParMETIS, MUMPS, and Zoltan.
- Most applications have chosen to implement fault-tolerance via their own checkpoint restart capability rather than rely on the need for a fault-tolerant communication library. Further possibilities in this regard should be pursued.
- Hybrid parallel programming models for efficient scaling on multicore processors need to be pursued vigorously.
- Large-scale application codes can easily have useful lifetimes of 20–50 years (corresponding to 5–10 generations of LCF systems), with the first 5–10 years (and ~100 person-years of effort) needed just to reach maturity. Expecting applications code developers to rewrite a mature code from scratch (e.g., in a new language) in order to achieve better scaling or parallel performance is therefore naïve. Applications code developers are talented; they are adept at and used to refactoring existing code to achieve better performance. This will be the approach of preference on petascale LCF systems. There is no magic language or compiler that can do better in this short time frame.
- With petascale LCF systems consisting of 100K (or more) nodes and/or processors, parallel algorithms must not only work, but their implementation must also conform to the highest

software quality assurance (SQA) standards. Software quality, and the breadth and depth of testing required to ensure and maintain this quality, is too often underemphasized or neglected under the pressure of producing timely science results. This trend could be exacerbated on LCF systems.

Runtime Footprint

- The path forward for many application areas includes either enhanced resolution or additional physics or both. This necessarily translates to increased aggregate and per-node memory requirements. Given the present cost of memory relative to processing power, this requirement represents a fundamental tension that must be carefully examined.
- Developer estimates for many code characteristics (e.g., memory usage, network bandwidth, wallclock time) are often misguided by poor implementations of algorithms and poor choices of software infrastructure (e.g., data structures). A basic understanding of fundamental algorithm characteristics (e.g., floating-point operations required, memory operations required) is necessary to accurately evaluate such requirements.

Data Analysis and Data Management

- I/O software packages and library requirements can be captured in a relatively small list.
- The NCCS LCF should consider procuring a cluster of "fat" nodes, each with multiple multicore processors and many gigabytes of memory. This fat-node cluster should have direct access to the local file systems on the LCF supercomputers, allowing efficient analysis, with standard tools such as IDL and Matlab, without having to move data over the WAN. Furthermore, for the visualization and analysis of the largest datasets, we believe that the aggregate memory of the cluster must be large enough to hold one full time step of state data.
- Developers should explore the use of asynchronous I/O, which could give the potential for decreasing output bandwidth requirements by an order of magnitude.
- Developers should explore performing analysis and data reduction within the parallel application itself, thus decreasing output bandwidth significantly and dramatically speeding analysis.

ACKNOWLEDGMENTS

The staff in the NCCS, particularly members of the Scientific Computing Group, would like to thank the computer and computational scientists involved in the INCITE Projects with allocation awards on the NCCS LCF in 2006 and 2007. Without their projects, applications, and time and insight responding to our requirements questions, this analysis would not have been possible. We would also like to thank the members of the ASCAC subpanel (in particular, chair Gordon Bell) who spent time with us and provided insight and guidance on science-based performance metrics for ASCR computational facilities. This work has been supported by the U.S. Department of Energy Office of Science Advanced Scientific Computing Research (ASCR) Program at Oak Ridge National Laboratory under contract DE-AC05-00OR22725 with UT-Battelle, LLC.

APPENDICES

Appendix A: Glossary of Application Codes

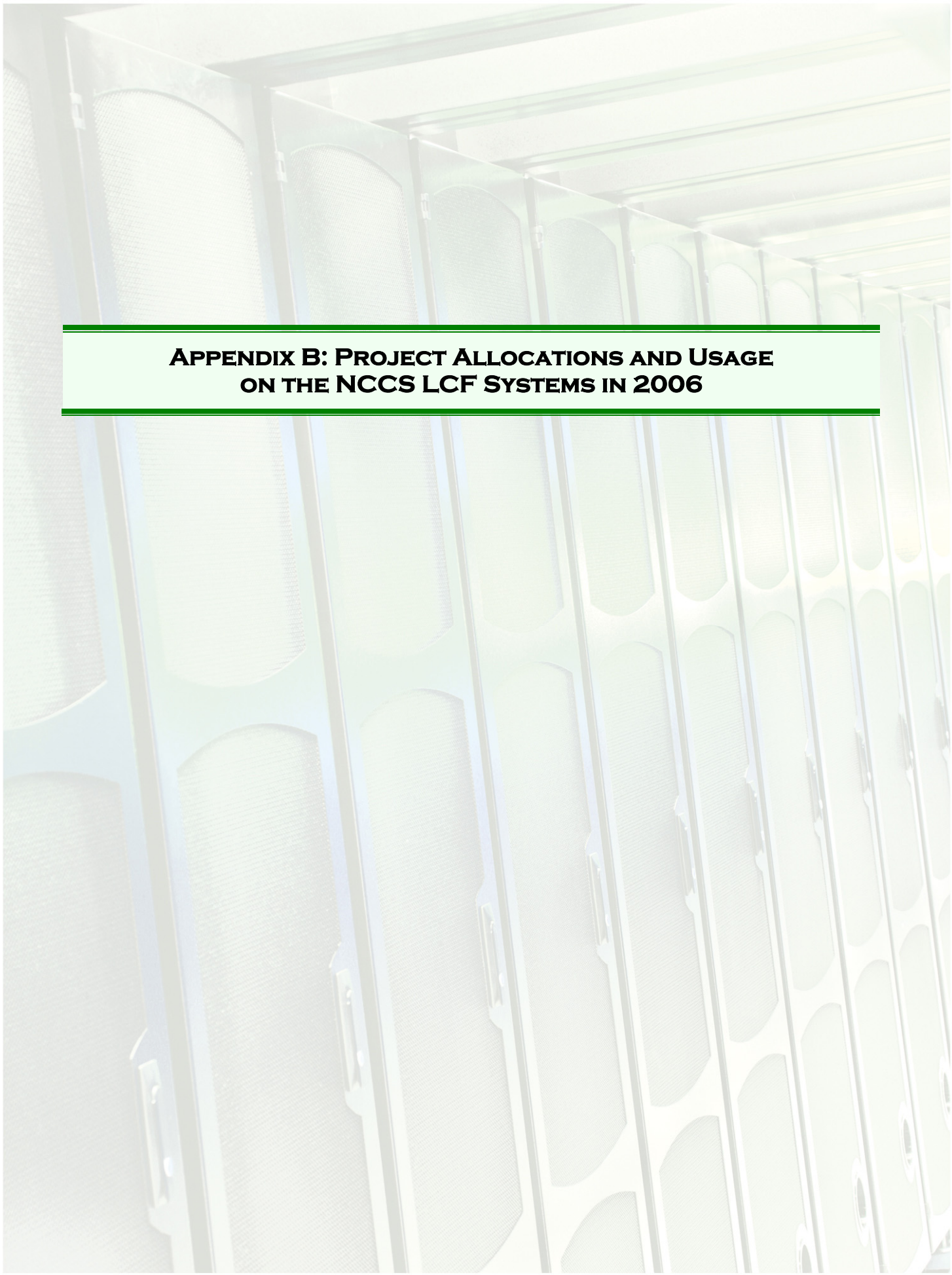
APPENDIX A: GLOSSARY OF APPLICATION CODES

The following is an alphabetical list of science application codes (by name) and the science domains in which they are used that have been installed and run on the NCCS LCF systems during the calendar year 2006. This list is not exhaustive, but it is representative of the codes possessed by the 22 projects (17 LCF and 5 INCITE projects) that received over 36 million processor hours of allocation on the NCCS Cray XT3 and Cray X1E LCF systems.

Active Harmony	Computer Science
ABinit	Chemistry
AMBER	Biology
AMRMHD	Fusion
AORSA	Fusion
BOLTZTRAN	Astrophysics
CASINO	Nanoscience
CCSM	Climate (Includes CAM, POP/CICE, CN, CASA, CLM)
CFL3D	Engineering
CHARMM	Biology
(m.b)CHIMERA	Astrophysics
CHROMA	High Energy Physics
CMS	High Energy Physics
CompHEP	High Energy Physics
CPMD	Chemistry
CQL3D	Fusion
DELTA5D	Fusion
ESPRESSO	Chemistry
FLASH	Astrophysics
FPMPI	Computer Science
GAMESS-US	Biology
GEM	Fusion
GeNASiS	Astrophysics
GEOS5	Climate
GTC	Fusion
GYRO	Fusion
HFB	Nuclear Physics

HPCC	Computer Science
HPCTOOLKIT	Computer Science
IPM	Computer Science
KOJAK	Computer Science
LAMMPS	Biology
LCG	High Energy Physics
LSMS	Materials and Nanoscience
MADNESS	Chemistry
M3D	Fusion
MILC	High Energy Physics
MITgcm	Climate
mpiP	Computer Science
NAMD	Biology
NIMROD	Fusion
NUCCOR	Nuclear Coupled Cluster — Oak Ridge
NWChem	Chemistry
OCTOPUS	Chemistry
Omega3P	Accelerator Physics
PAPI	Computer Science
PARADYN	Computer Science
PMaC	Computer Science
PWSCF	Chemistry
PYTHIA	High Energy Physics
QBOX	Materials Science
QMC/DCA	Materials & Nanoscience
ROOT	High Energy Physics
ROSE	Computer Science
S3D	Combustion
S3P	Accelerator Physics
SMMC	Nuclear Physics
SPF	Materials and Nanoscience
SUPERNOVA	Astrophysics
SvPablo	Computer Science
T3P	Accelerator Physics

TAU	Computer Science
TORIC	Fusion
V2D	Astrophysics
VASP	Materials and Nanoscience, Chemistry
VH1/EVH1	Astrophysics
VULCAN/2D	Astrophysics
WRF	Climate
XGC-ET	Fusion
ZEUS-MP	Astrophysics



Appendix B: Project Allocations and Usage on the NCCS LCF Systems in 2006

APPENDIX B: PROJECT ALLOCATIONS AND USAGE ON THE NCCS LCF SYSTEMS IN 2006

Table B.1 summarizes the 22 projects awarded allocations on the NCCS LCF systems during the calendar year 2006: 17 of these projects were LCF projects, with the stipulation that the principal investigator's (PI's) research be funded by DOE, and the remaining 5 project were INCITE projects, which were broadly open to PIs from industry and academia, whose research was not necessarily funded by DOE.

Table B.1. Projects receiving allocation awards on the NCCS LCF systems in 2006

Science domain	Project title	Project type	Principal investigator	Cray XT3 allocation	Cray X1E allocation
Astrophysics	*Multi-dimensional Simulations of Core-Collapse Supernovae*	LCF	Adam Burrows	1.25M 4.1%	0 0.0%
Astrophysics	*Ignition and Flame Propagation in Type la Supernovae*	LCF	Stan Woosley	3.00M 9.9%	0 0.0%
Astrophysics	*Multi-dimensional Simulations of Core-Collapse Supernovae*	LCF	Tony Mezzacappa	3.55M 11.7%	0.70M 11.9%
Biology	*Next Generation Simulations in Biology*	LCF	Pratul Agarwal	0.50M 1.7%	0 0.0%
Biology	*Molecular Dynamics Simulations of Molecular Motors*	INCITE	Martin Karplus	1.48M 4.9%	0 0.0%
Chemistry	*Rational Design of Chemical Catalysts*	LCF	Robert Harrison	1.00M 3.3%	0.30M 5.1%
Climate	*Role of Eddies in Thermohaline Circulation*	LCF	Paoli Cessi	0 0.0%	0.03M 0.5%
Climate	*Climate-Science Computational End Station*	LCF	Warren Washington	3.00M 9.9%	2.00M 33.9%
Climate	*Studies of Turbulent Transport in the Global Ocean*	LCF	Synte Peacock	1.50M 4.9%	0 0.0%
Computer Science	*PEAC End Station*	LCF	Patrick Worley	1.00M 3.3%	0.20M 3.4%
Computer science	*Real-Time Ray-Tracing*	INCITE	Evan Smyth	0.95M 3.1%	0 0.0%
Materials science	*Simulations in Strongly Correlated Electron Systems*	LCF	Thomas Schulthess	3.50M 11.6%	0.30M 5.1%
Engineering	*Large Scale Computational Tools for Flight Vehicles*	INCITE	Moeljo Hong	0 0.0%	0.20M 3.4%

Table B.1 (continued)

Science domain	Project title	Project type	Principal investigator	Cray XT3 allocation	Cray X1E allocation
Materials science	*Numerical Simulation of Brittle and Ductile Materials*	INCITE	Michael Ortiz	0.50M 1.7%	0 0.0%
Fusion	*Gyrokinetic Plasma Simulation*	LCF	W. W. Lee	2.00M 6.6%	0.23M 3.8%
Fusion	*Tokamak Operating Regimes Using Gyrokinetic Simulations*	LCF	Jeff Candy	0 0.0%	0.44M 7.5%
Fusion	*Wave-Plasma Interaction and Extended MHD in Fusion Systems*	LCF	Don Batchelor	3.00M 9.9%	0 0.0%
Fusion	*Interaction of ETG and ITG/TEM Gyrokinetic Turbulence*	INCITE	Ronald Waltz	0 0.0%	0.40M 6.8%
High energy physics	*Reconstruction of CompHEP-produced Hadronic Backgrounds*	LCF	Harvey Newman	0.03M 0.1%	0 0.0%
Accelerator physics	*Design of Low-loss Accelerating Cavity for the ILC*	LCF	Kwok Ko	0 0.0%	0.50M 8.5%
Nuclear physics	*Ab-inito Nuclear Structure Computations*	LCF	David Dean	1.00M 3.3%	0 0.0%
Combustion	*High-Fidelity Numerical Simulations of Turbulent Combustion*	LCF	Jackie Chen	3.00M 9.9%	0.60M 10.2%

During the 9-month period from January to September 2006, the science application codes associated with the 22 LCF and INCITE projects executed on the NCCS LCF Cray XT3 (Jaguar) and Cray X1E (Phoenix) resources according to the job size distribution shown in the bar chart in Fig. B.1 and Table B.2. Job size is defined as the number of processors utilized during any given calculation (placed in discrete bins), the percentage of utilization for a job distribution bin is defined as the total number of processor hours utilized in that job size range (e.g., 129–256 nodes) divided by the total number of processor hours used on that particular system over the time period of interest (Fig. B.2). Note that both of the LCF systems have a job distribution load characteristic of a leadership (or capability) usage model, namely, skewed heavily toward usage of a large percentage (e.g., >25%) of the total available resource for any given calculation. Also shown in Figs. B.3–B.6 is month-to-month evolution of job distribution and utilization on each LCF system.

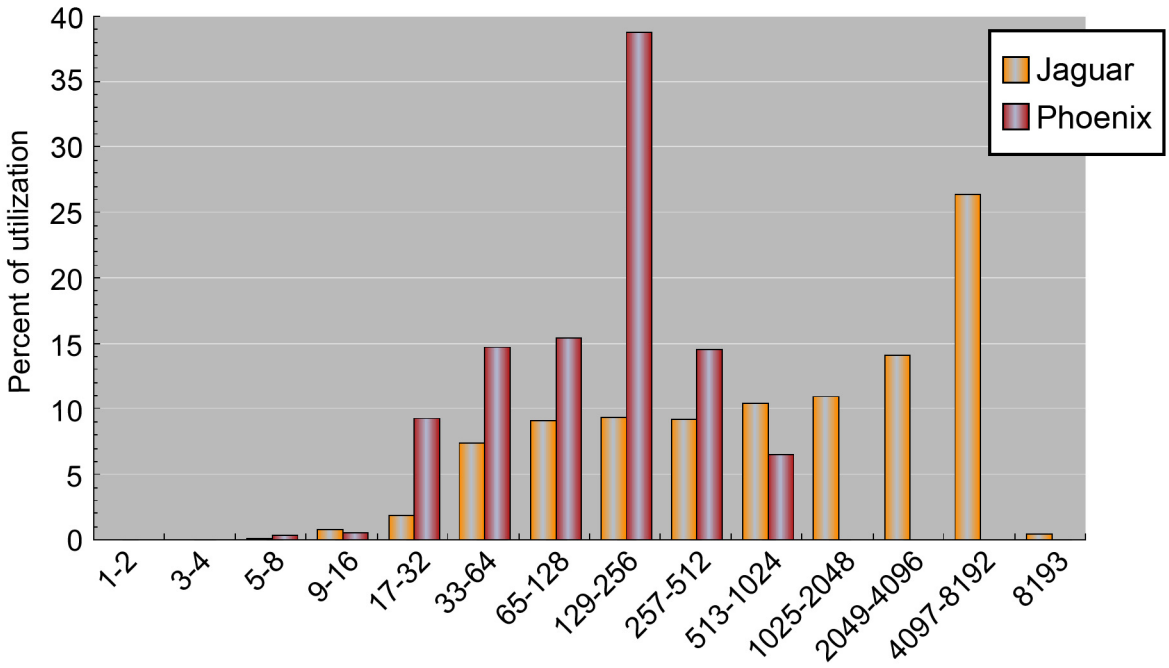


Fig. B.1. Job size distribution of allocated science applications on the NCCS LCF systems in the January to September 2006 time period. Shown is the percentage of total utilization as a function of calculation size (number of processors).

Table B.2. Job size distribution of allocated science applications on the NCCS LCF systems in the January–October 2006 time period

Job Size (processor count)	Jaguar (Jan 2006–Oct 2006)		Phoenix (Jan 2006–Oct 2006)	
	Fraction of total system (%)	Fraction of total utilization (%)	Fraction of total system (%)	Fraction of total utilization (%)
1–2	<0.04	0.01	<0.2	0.02
3–4	<0.08	0.02	<0.4	0.05
5–8	<0.2	0.11	<0.8	0.39
9–16	<0.3	0.92	<1.6	0.89
17–32	<0.6	2.73	<3.1	10.15
33–64	<1.2	9.99	<6.3	24.87
65–128	<2.5	18.96	<12.5	40.65
129–256	<4.9	28.40	<25.0	79.26
257–512	<9.8	37.43	<50.0	93.58
513–1024	<19.7	47.69	<100.0	100.0
1025–2048	<39.3	58.39	N/A	N/A
2049–4096	<78.6	72.46	N/A	N/A
4097+	<100.0	100.0	N/A	N/A

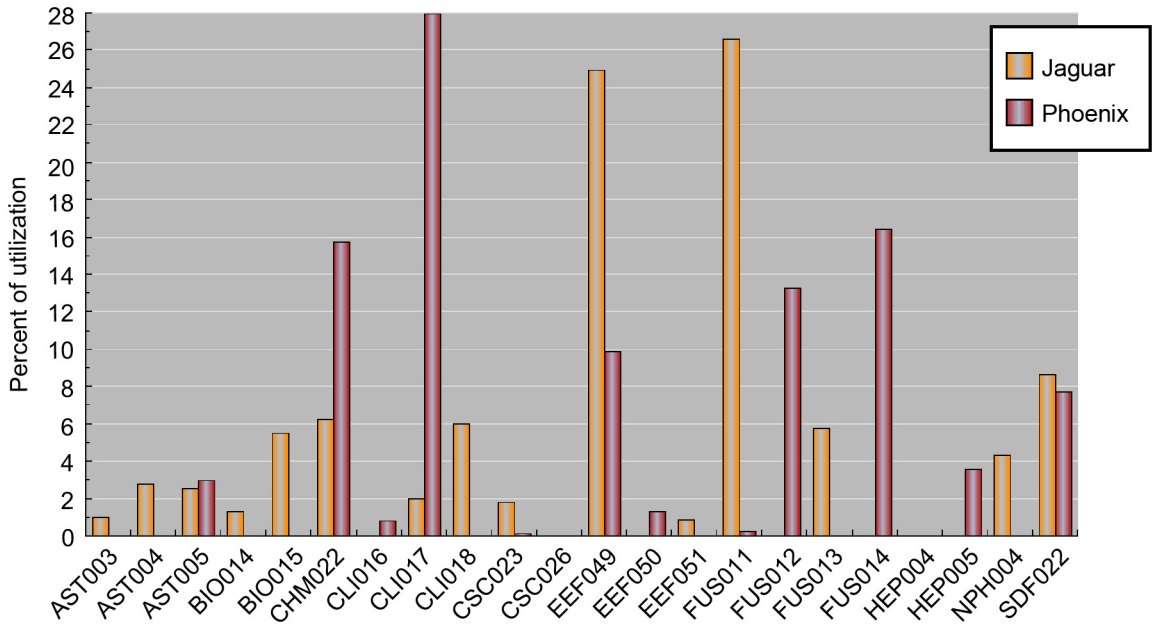


Fig. B.2. Percentage of total NCCS LCF system utilization for each project receiving a 2006 allocation award in the January to September 2006 time period. Shown is the percentage of total utilization as a function of calculation size (number of processors).

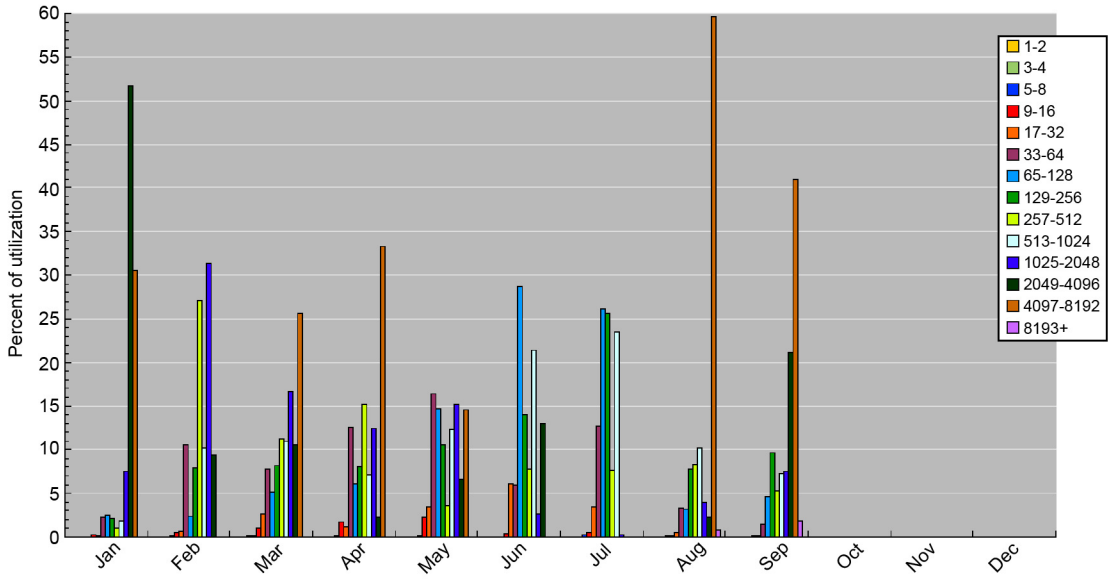


Fig. B.3. Month-by-month change in the job size distribution of the allocated science applications on the NCCS Cray XT3 (Jaguar) system in the January to September 2006 time period. Shown is the percentage of total utilization as a function of calculation size (number of processors).

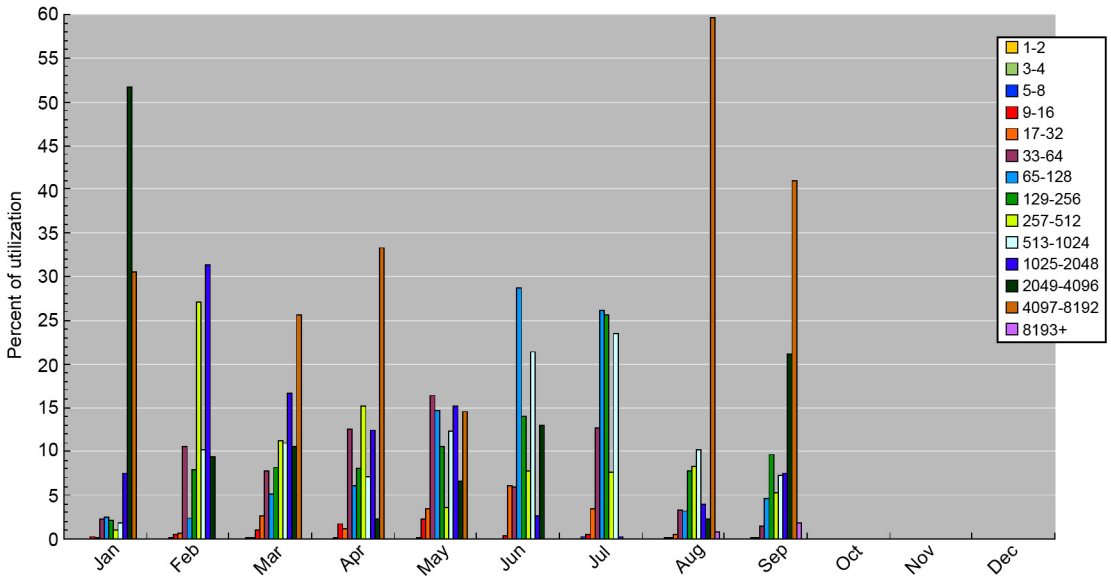


Fig. B.4. Month-by-month change in the job size distribution of the allocated science applications on the NCCS Cray X1E (Phoenix) system in the January to September 2006 time period. Shown is the percentage of total utilization as a function of calculation size (number of processors).

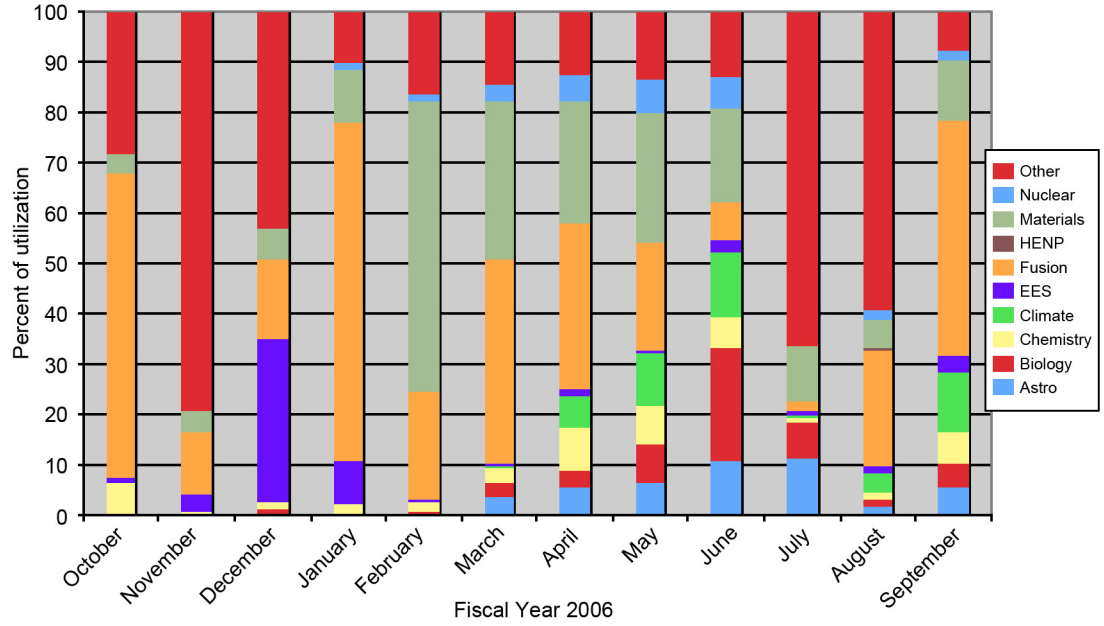


Fig. B.5. FY 06 utilization on the Cray XT3 (Jaguar) system by scientific discipline.

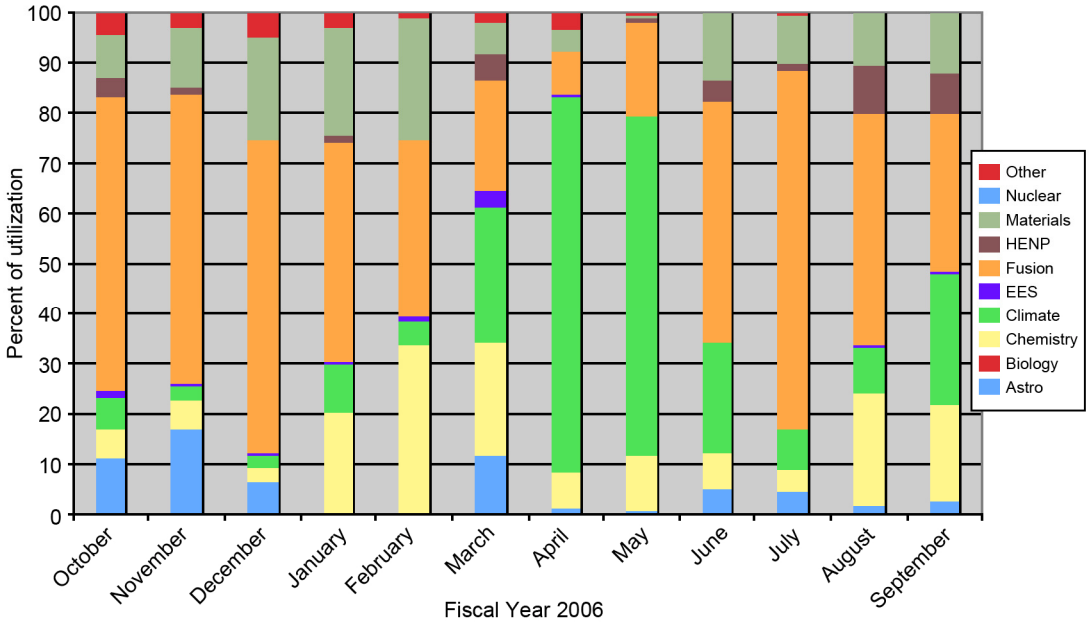


Fig. B.6. FY 06 utilization on the Cray X1E (Phoenix) system by scientific discipline.

As expected, the distribution of jobs over the course of the year tends to reflect an evolving leadership usage distribution (i.e., skewed more toward capacity early in the year when the applications are more immature and not as adept with regard to scalability). As the parallel algorithms are tuned, optimized, and in some cases redesigned, the applications make more effective use of the full resource, and by the end of the year the job distribution tends to skew toward fuller utilization of the LCF resources.

Appendix C: Applications Requirement Council

APPENDIX C: APPLICATIONS REQUIREMENT COUNCIL

The NCCS requirements process—the development, management, and planning of requirements for NCCS clients, customers, and users (the "stakeholders")—is the responsibility of the Applications Requirements Council (ARC). The ARC, whose charter is given below, is the NCCS requirements board in a formal project-management sense. ARC requirements are passed on to the NCCS Technology Council (TC), where they serve as guidance, constraints, and specifications for the upgrade, selection, procurement, acceptance testing, and even design of current and next-generation leadership systems.

The ARC develops, manages, and plans the breakthrough science requirements imposed upon the NCCS leadership computing systems. These requirements are embodied within the simulation tools used, developed, and envisioned by scientists pursuing these tools as vehicles for discovery, exploration, and validation of their research. The principal product of the ARC is the documentation, publication, and handoff of requirements to the NCCS TC, which is responsible for implementing and/or aligning these requirements with deployed NCCS leadership computing systems. By articulating requirements, the ARC will help to ensure that all NCCS systems are aligned to the maximum extent possible with the needs and goals of the breakthrough science projects using the NCCS resources. ARC requirements apply to the entire end-to-end analysis process followed by scientists using the NCCS facilities. The process includes system hardware, system software, the integrated development environment, and the problem-solving environment, which includes data analysis, management, and visualization. It is our vision that the ARC will positively influence the design, procurement, deployment, and/or operation (e.g., user and technical support) of an NCCS system by improving the quality, quantity, or fidelity of the output of one or more breakthrough science simulation applications in a measurable way.

The remainder of this section gives the process for the development, management, and planning of NCCS stakeholder requirements. This process translates into explicit ARC tasks and milestones, many of which are repeated on an annual basis. The ARC requirements process follows three basic steps: development, management, and planning.

Requirements Development

Requirements are developed after (in sequence) elicitation, analysis, validation, and specification.

Requirements Elicitation

Requirements elicitation was performed by reformulating the four basic types of requirements (business, functional, quality, and design) into a series of questions to ask specific people, who were in one of three categories—client, customer, or user. Clients are those who pay for product development; customers pay for the product; and users use the product. The next task was to identify people to

interview (preferably at least one per category) and formulate several questions per requirement. We then sought answers to these questions from the selected people through e-mail, phone calls, meetings (interviews), or workshops. The answers were then documented and verified by the interviewee to ensure that the answers correctly reflected the opinion of the person interviewed. We found that using models during the interview and encouraging the interviewee to change them was an effective method for clarifying responses. Asking negative questions was an effective means for soliciting quantitative answers. The elicitation process is a two-step process: (1) identify the stakeholders and (2) interview the stakeholders.

The following questions are suggested as guidance in the elicitation process:

SCIENCE MOTIVATION AND IMPACT

- Why does your science need leadership computing? Without leadership computing, can progress be made at all? Or as fast?
- What science questions are you answering? When are these answers needed? Why are these answers needed? What are the impacts of having the answers to these questions? Will having those answers mean you are finished or lead to new questions?
- Is your science to be validated against observation (experimental validation) or solely the instantiation of theory?
- Who are your clients (who pays for product development)?
- Who are your customers (who pays for the product)?
- What is your product? Answers to fundamental understanding or to guide theory? Will the software be released to others? Will you provide guidance and/or optimization in designing an experiment or actual end product? All of the above?

SCIENCE QUALITY AND PRODUCTIVITY

- How might the quality (fidelity of physics models) of your science quality change with system peak speed and aggregate memory?
  - 25 TF
    - List of physics models currently in your application
    - Specify the time and length scales resolved (if appropriate)
  - 100 TF, 250 TF, 1 PF, sustained PF
    - List of new and improved physics models now possible
    - Specify the time and length scales resolvable (if appropriate)

- How might the productivity of your science output change with system peak speed and aggregate memory? Is this important?
  - 25 TF
    - Estimate the time with your current simulation tool that it takes to arrive at a simulation-informed decision, discovery, experimental design, or training/education exercise
  - 100 TF, 250 TF, 1 PF, sustained PF
    - Estimate time improvement
- What maximum simulation turn-around time can you tolerate and still move your science forward? What turn-around time would allow detailed parameter studies and optimization?
- What are the current time-intensive bottlenecks for your workflow process? What might this process be for you in 5 years (e.g., with >1 PF)?
- Can you give a use case for today's resources? A use case is characterized by workflow (problem definition, problem setup, main compute phase, post processing, data analysis and visualization, dissemination of results), reason for simulation, recipient of the result, physics/algorithm aspects of the simulation. How might a use case look in 5 years?
- How many instances of use cases are required for scientific discovery?
- Are your computational experiments sequential (the current dependent upon the previous result)? How many experiments might be performed in a year?
- Can your simulations be validated (physical models compared against experimental data)? If so, have they been? To what extent (breadth, depth, and quality) has your simulation tool been validated?
- What confidence level (level of predictability) do you have in your current simulations? Can this be quantified (e.g., "error bars")? If not, is this possible with more computational resources? What physics models are crudely represented today (have the highest uncertainties)? How might this change in 5 years (with >1 PF)?

APPLICATION MODELS

- How many state variables currently describe your physical system? How might this change in 5 years (with >1 PF)?
- Are your models deterministic? Stochastic? Both? If deterministic, how are your models expressed (e.g., PDEs)? How might this change in 5 years (> 1 PF)?
- Are multiple, simultaneous physical processes modeled? If so, are fully coupled solutions obtained? How might this approach change in 5 years (> 1 PF)?

- Is the domain of dependence for any given state variable local (dependent upon other nearby state variables) or global?
- Is your physical model data dependent, meaning that the actual physical model invoked dynamically depends upon data itself (e.g., an interfacial physics model is only invoked along interfaces)?

Application algorithms

- What is your current parallelism model (distributed, domain replicated, etc.)? Are you instantiating parallelism with MPI tasks, threads, or both?
- We are currently at 5K threads of execution. Do you have any algorithms that may not scale to 100K threads of execution? If so, why and what are the obstacles?
- Have you been able to quantify numerical errors and convergence properties of your algorithms? How might this change on larger systems?
- Do your algorithms require solutions to linear and/or nonlinear sets of equations? If so, are these local, global, sparse, dense? How might this change in 5 years (> 1 PF)?
- Do your algorithms adaptively change as a function of space and time, based on the data (e.g., AMR)? How might this change in 5 years (> 1 PF)?

Application software

- What programming languages and external libraries, and tools (compilers, debuggers) do you require? How might this change in 5 (> 1 PF) years?
- How are your simulations verified (solving equations correctly)?
- What kind of testing (e.g., unit, regression, integral, etc.) do you perform? Is the breadth, depth, and quality of your software and algorithm verification testing adequate?
- What development tools (IDE, editors, compilers, debuggers) do you require?
- What is the biggest time bottleneck in the IDE cycle?
- Is your software under active development? If so, by a single individual or a team? Are software engineering and software project management best practices found useful and followed?
- How would you rate the quality and maturity of your software?

Application footprint on the system (hardware)

- What is your current I/O model (parallel, serial through a single PE)?
- What are the frequency and size (in terms of fraction of simulation image) of your restart and graphics dumps?
- Does your application require an extensive amount of indirect addressing?

- Can your application execute on a heterogeneous system? If not, would the workload be amenable to this?
- Does your application require dynamic repartitioning? Might it in 5 years (>1 PF)?
- Is your application load balanced?
- What are your application communication needs in terms of locality and regularity? How might this change in 5 years (>1 PF)?
- Does your application have a few identifiable performance bottlenecks? Are these bottlenecks localized in software?
- Do you have a normalized performance metric for your application (e.g., grind time)? If so, what is it and is it being tracked? What fraction of peak system speed is being realized? What fraction of the total cycles is devoted to floating point ops, integer ops, logical ops, data movement, etc.?
- What is your application's normalized memory usage (e.g., double precision words required per discrete solution point or cell)? What fraction of this can be accounted for by the permanent state variables representing the physical system you are modeling?

Data management and analysis

- What analysis tools (data mining, visualization, etc.) do you require?
- For a typical leadership simulation, what is your temporary and archival storage size needs (expressed as a function of the simulation image size)?
- For a typical leadership simulation, what are your needs for maximum allowable read/write times to temporary storage (expressed as a function of the total simulation time)?
- Why do you need archival storage? Are your simulation datasets analyzed and used by many others or are they for single-user backup?

Miscellaneous

- What keeps you awake at night about your simulation tool?
- What are the highest risks or impediments to success?

Analysis

Analysis is the process by which gaps or missing requirements are identified, which includes possibly new elicitation, negotiation of scope, and establishment of consistency. Models such as state diagrams, information/class diagrams, and data flow diagrams are used to analyze requirements. Analysis also involves prioritization (based on issues like scale, cost, benefit, risk) of existing requirements; this information is often obtained by interviewing the key players once again. Prioritization schemes based on

numerical scales (1–10), enumerated scales, or timelines can be used. In setting priorities, a forced distribution should be used: 25% in the top tier, 50% in the middle tier, and 25% in the low tier.

Validation

Validation is determining whether the requirements are "good enough" (attained) as well as being the correct set (allocation). Requirements are validated through peer reviews, test case creation, and target/estimation alignment. Good requirements must be unambiguous, testable, correct, in scope, modifiable, feasible, traceable, written in clear (customer's) language, acceptable to all clients, and not a solution.

Specification

Requirements can be specified in templates (text or models), user manuals, test cases, or prototypes.

Requirements Management

Requirements are managed by scrubbing, tracking change, and matching scope.

Scrubbing

Specified requirements are scrubbed by eliminating those that are not "important" and simplifying those that are unnecessarily complicated. Scrubbing occurs via a requirements review (peer review) board, which is currently the ARC.

Change Management

Changes in requirements must be documented, tracked, and approved, just like change control of any software. The ARC serves as the change board.

Scope Matching

In considering changes to requirements, attention must be paid to the impact the change has on benefits, cost, risk, schedule, quality, resource allocation, timing, and stability.

Requirements Planning

Since requirements development and management is an ongoing process, the necessary supporting activities must be planned ahead of time. A requirements plan is the documentation of the process by which requirements are developed and managed. The ARC, chaired by the NCCS Director of Science, owns and executes requirements development and management.

ARC Implementation

ARC membership consists of the NCCS Director of Science (co-chair), the group leader and staff of NCCS Scientific Computing (SC), and external users and scientists selected by the ARC chair to represent each domain science (at least one member per domain) currently supported by NCCS resources. One external member will serve as co-chair along with the Director of Science. ARC membership will be for one year with ongoing renewals acceptable. The ARC shall communicate regularly through monthly teleconference calls, face-to-face group meetings (at least one annually), electronic mailing lists, sharepoint Web sites, one-on-one meetings, phone conversations, and e-mail messages.

ARC Tasks and Milestones

To develop, manage, and plan applications requirements, the ARC will

- Maintain regular phone conversations and e-mail exchanges about requirements issues among its members (NCCS Director of Science, SC group staff, and LCF INCITE project points of contact),
- Conduct regular (every four to six week) teleconferences to develop, manage, and plan requirements,
- Keep abreast of other related computational science requirements management efforts and any associated documentation, workshops, and activities,
- Document and publicize an annual applications requirements document,
- Hold an annual (end of fiscal year) PI meeting where LCF and INCITE projects discuss and present their science results as well as current and anticipated requirements,
- Hold at least one face-to-face meeting annually (e.g., at the users or PI meeting),
- Produce quarterly LCF and INCITE project updates on science results and requirements,
- Maintain a sharepoint Web site, and
- Provide formal requirements-based technology recommendations to the TC that are as actionable as possible since the TC helps to design, procure, and deploy computer systems that best meet the requirements submitted by the ARC.

The functions, actions, and outcomes of the ARC are the responsibility of the NCCS Director of Science.

Appendix D: ASCAC Code Project Questionnaire

APPENDIX D: ASCAC CODE PROJECT QUESTIONNAIRE

The following is the project survey template developed by members of the ASCAC subpanel on science-based performance metrics for ASCR computational facilities. These surveys were used by members of the NCCS Scientific Computing Group in gathering requirements-relevant information from each LCF and INCITE project teams receiving an allocation on the NCCS LCF systems in 2006.

Experiment Project Overview

- Project name
- Contact information for the project
  - Principal investigators, e-mails, phones
  - URL
- DOE Office support: DOE program manager; SC Office (BES, BER, NP, HEP, ASCR, FES, other)
- Scientific domain (chemistry, fusion, high energy, nuclear, other)
- What are the technical goals of the project?
  - What problem or "grand challenge" are you trying to solve?
  - What is the expect impact of project success? (e.g., better understanding of supernova explosions, prediction of ITER performance)?
- Support for the development of the code
  - Degree of DOE support to develop the code
  - Other agency support
- What is the project profile in total human resources, including
  - Trained scientists
  - Program development and maintenance
  - User(s) of the team codes
- Size of any or all external communities that your code or datasets support:

Project Team Resources

- Team size
- Team institutional affiliation(s)
- To what extent are the code team members affiliated with the computer center institution? Team composition and experience total
  - Domain scientists

- Team composition by educational level (total)
  - Ph.D.
  - M.S., B.S., undergraduate students, graduate students
- Team resources utilization: time spent on code and algorithm development, maintenance, and problem setup, production, and results analysis

PROJECT CODE

- Problem type (data analysis, data mining, simulation, experimental design, etc.)
- Types of algorithms and computational mathematics (e.g., finite element, finite volume, Monte-Carlo, Krylov methods, adaptive mesh refinement, etc.)
- What systems does your code run on?
- What is your preferred system?
- Code size (single lines of code, function points, etc.)
- Code age
- Amount of code added per year
- Computer languages employed
  - Fortran
  - C
  - C++
- Structure of the codes
  - What libraries are used?
  - What fraction of the effort do they represent?
- Code mix
  - To what extent does your team develop and use your own codes?
  - Codes developed by others in the DOE and general scientific community?
- What is the present parallel scalability?
  - Projected or maximum scalability?
  - How is measured?
  - Is the code massively parallel?
- What memory/processor ratio does your project require? (i.e., gigabytes/processor)?
- Parallelization model
  - Does your team use domain decomposition and if so what tools do you use?
- What is the "efficiency" of the code?
  - How is it measured?

- What are the major bottlenecks for scaling your code?
- What is the split between interactive and batch use?
- What is the split between code development on the computer center computers and on computers at other institutions?

Project resources input from the centers

- Steady state user of resources on a production basis per month
  - Processor number
  - Processor time
  - Disk
  - Tertiary rate of change
- Annual use of resources
  - Processor time
  - Disk
  - Tertiary storage rate of change
- Software provided by center
- Consulting
- Direct project support as a team member
- What is the size of the job in terms of memory, concurrency (processors), disk, and tertiary store?
- What is the scalability of these codes?
- What is the wall-clock time for typical runs?

Software Engineering, Development, Verification and Validation Processes

- Software development tools used
  - Parallel development
  - Debuggers
  - Visualization
  - Production management and steering
- Software engineering practices. Please list the specific tools or processes used for:
  - Configuration management
  - Quality control
  - Bug reporting an tracking
  - Code reviews

  - Project planning
  - Project scheduling an tracking
- What is your verification strategy?
- What use do you make of regression tests?
- What is your validation strategy?
- What experimental facilities do you use for validation?
- Does your project have adequate resources for validation?

Project output (t) and user metrics

- Enumerate project output, consisting mainly of journal publications, dissertations, and research reports
- In addition provide: number of publications, citations, dissertations, prizes and other honors
- Residual and supported, living datasets and/or databases that are accessed by a community
- Describe size of the external user community for the datasets
- Change in code capabilities and quality (t)
- Code contributed to the centers
- Code contributed to the scientific community at large
- Company spin-offs based on code or trained people and/or CRADAs
- Corporation, extra-agency, etc. use
- Increase in trained scientists during 2001–2005
- Increase in trained code developers capable of writing project-level codes during 2001–2005

Project Future (qualitative)

- What is today's greatest impediment in terms of your use of the center's computational facilities?
- With the projected increases resources over next 3 years?
- What do you believe the proposed increases in capacity at the facilities will provide (e.g., based on observations of historical increases)?
  - Better turn-around time for the project
  - More users and incremental improvement in use with little or no change in scale or quality
  - Reduced granularity, resulting in constant solution time, though more accurate results
  - New applications permitting in new approaches and new science
  - How, specifically, has your use changed with specific facilities increases
- How is the project effort projected to change in the next 5 years?
- What is your plan for utilizing increased resources?

Appendix E: Survey of Acceptance and Early Access Science Applications

APPENDIX E: SURVEY OF ACCEPTANCE AND EARLY ACCESS SCIENCE APPLICATIONS

This appendix contains a representative list of science application codes considered to be priority petascale applications. These codes are suitable for acceptance testing as well as possessing high potential for achieving breakthrough science results. The science outlined by each application would benefit greatly from early access to "science at scale" simulation time on the planned 250-TF and 1000-TF Leadership Computing Facility (LCF) systems at Oak Ridge National Laboratory. This candidate list, which is not exhaustive, represents what we believe to be an excellent set of codes that have high potential of scaling to fully utilize the 250-TF and 1000-TF systems while achieving breakthrough science with the resulting simulations. The codes span many domains of science and a wide variety of models, algorithms, and software that collectively stress all aspects of a petascale computational resource. These applications originate from many different institutions. For each code in this list the following are summarized: physical models, numerical algorithms, current and project scaling performance, ways to be used in an acceptance test, the science it might probe with early access simulations, its functional software requirements (system software and math libraries), and points of contact. The code data is based in large part on details graciously provided by the relevant code authors and subject matter experts. The authors of this document have attempted to compile a complete list of available codes in Table E.1. Details for each code are provided in the text following the table.

Table E.1. Acceptance test utility, description, and metrics for selected science application codes

Code	Acceptance test utility	Acceptance test description and metrics
AORSA	Scalability; functionality	Scale up on full system with problem having a known answer; test complex factorization of dense matrices (ScaLAPACK/PBLAS)
CAM	Single PE performance	Test problem of finite volume dynamical core with atmospheric chemistry
CASINO	Single PE performance	Perform a 10000 electron system calculation
NUCCOR	Functionality	Reproduce regression tests
CHIMERA	Functionality; interconnect B/W	Standard explicit hydro test problem with a known answer (Sod)
FLASH	Scalability; I/O	Reproduce isotropic DNS run on BG/L; whole star type 1A simulation, ensuring amount of burned mass at same resolution is identical to a prior result obtained on another system; write out restarts with <10% overhead
GTC	I/O; scalability	Write out restarts with <1% of overhead; push 20B particles in one step in one second

Table E.1 (continued)

Code	Acceptance test utility	Acceptance test description and metrics
LAMMPS	Functionality	~1M atom simulation generating reproducible answers
LSMS (+WL)	Scalability; stability	Use a simple, well known bulk system (bcc Fe) to test stability and scaling of message-passing performance
MADNESS	Functionality	Compile and run code with correct answers
MILC/CHROMA	Scalability	>15% of peak performance realized in the conjugate gradient portion of standard MILC without optimization
NEWTRNX	Functionality	Reproduce regression tests
NWChem	Functionality	Test proper implementation of global arrays
PFLOTRAN	Functionality	Exercises a large portion of the PETSc code base
POP/CICE	Single PE performance; interconnect latency	Runtime in simulated years per CPU day for a fixed-size problem (0.10 degree)
QBOX	Scalability	Parallel efficiency; percent of peak
QMC/DCA	Functionality	Generate test runs of virtually any size with a known answer that checks MPI, BLAS, LAPACK, and F90 compiler
S3D	Interconnect latency; I/O	Simulation with production I/O works and generates correct answers; runs correctly at a variety of PE counts; flame benchmark (time/step/grid point) is constant or better
T3P	Functionality	Test functionality of Zoltan, MUMPS, ScaLAPACK
VASP (+WL)	Functionality	Test functionality of BLAS, ScaLAPACK

AORSA

Physics Models

The All-Orders Spectral Algorithm (AORSA) code solves Maxwell-Boltzmann equations for the wave electric and magnetic fields and for the distribution function $f_s(r, v, t)$, representing the density of species in a 6-D phase space. The time-evolution of this function is determined using self-consistent electric and magnetic fields. The wave fields and particle distribution function can be separated into a time-averaged slowly varying part, (E_0, B_0, f_s^0), and a time harmonic rapidly oscillating part, [$E(r)e^{-i\omega t}$, $B(r)\,e^{-i\omega t}$, $f_s^1(r,v)e^{-i\omega t}$] where ω is the frequency of the wave. Solving the linearized Boltzmann equation gives the rapidly varying part of the distribution function $f_s^1(r, v)$ in terms of the equilibrium part f_s^0. For the rapidly oscillating, time harmonic wave fields, Maxwell's equations reduce to a generalization of the Helmholtz wave equation. The numerical solution is expensive because of the nonlocal nature of the plasma current, the geometric complexity of the plasma boundary, and the enormous range of spatial scales that must be treated. AORSA takes advantage of today's parallel computers and solves its equations in the general integral form with no restriction on wavelength relative to orbit size and no limit

on the number of cyclotron harmonics. AORSA has been generalized to treat nonthermal (i.e., non-Maxwellian) plasma components.

Algorithms

AORSA uses a fully spectral method to solve the wave equation and the resulting set of linear equations is solved using ScaLAPACK libraries or HPL, modified for use with complex coefficient systems (E. F. D'Azevedo et al., "Complex Version of High Performance Computing Linpack Benchmark [HPL]," SIAM Conference on Computational Science and Engineering, 2007). This avoids complicated convolutions associated with calculating the plasma current, and at the same time, includes cyclotron harmonics of arbitrarily high order. For an N × N grid in 2-D, AORSA generates a dense matrix of approximately $0.70*(3*N^2)$. For example, the medium-size ITER problem (128 × 128) requires the solution of a double complex valued linear system of order 34,692. The larger ITER problem (256 × 256) required to resolve the mode-converted waves requires solution of a linear system of order 124,587.

Scaling

Linear scaling up to 4096 processors; prefer twice the memory of Jaguar's processors (2 Gbytes/processor available to code); domain decomposition with MPI; 50% of peak on Jaguar; and 757 GF on 2024 Seaborg processors (Fig. E.1).

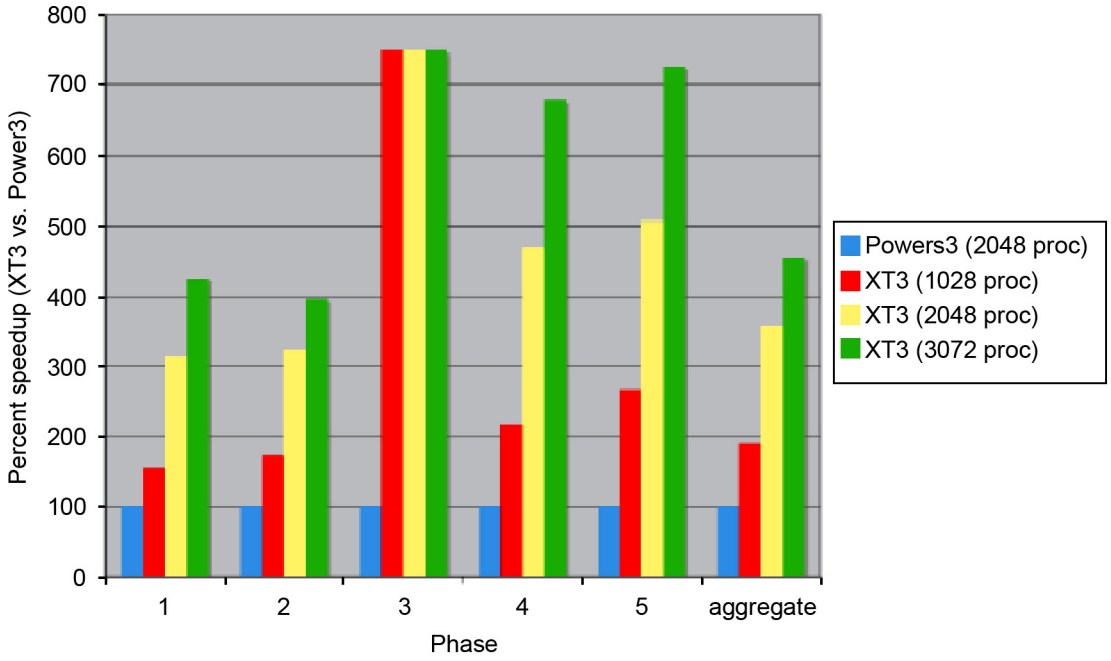


Fig. E.1. AORSA on the Cray XT series Jaguar system compared with an IBM Power3. The columns represent execution phases of the code. The aggregate is the total wall time, with Jaguar showing more than a factor of 3 improvements over Seaborg.

If Chosen for Science Day One

On day one with a petaflop, AORSA could do a complete simulation of mode conversion heating in ITER with a realistic antenna geometry and non-Maxwellian alpha particles. Right now, it takes 5000 processor hours to simulate mode conversion for a single toroidal mode with Maxwellian alphas. Non-Maxwellian alphas would take three times as long (15,000 processing hours) and then ten nonlinear iterations of this case with CQL3D (150,000 processor hours.) Then the realistic antenna would take about 100 toroidal modes or 15,000,000 processing hours.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Fortran 77/90	BLACS	NetCDF	None

Libraries and Tools

Library	Function	Functionality
SCALAPACK	PZGETRF	Double complex factorization of a dense matrix
SCALAPACK	PZGETRS	Double complex triangular matrix solve
PBLAS	PZSCAL	Double complex scaling of a submatrix
PBLAS	PZGECOPY	Double complex copying of a submatrix
FFTPACK	ZFFTF, ZFFTB, ZFFTI	Double complex forward, backward, and initialization
BLACS	ZGSUM2D	Double complex reduction sum of a submatrix
BLACS	BLACS_BARRIER	Interprocess barrier

Code Reference

Fred Jaeger (jaegeref@ornl.gov)

E. F. Jaeger et al., "Self-Consistent Full-Wave and Fokker-Planck Calculations for Ion Cyclotron Heating in Non-Maxwellian Plasmas," *Physics of Plasmas* **13**, 056101 (2006).

NCCS Point of Contact

Richard Barrett

rbarrett@ornl.gov

CAM

Physics Models

The general circulation of the atmosphere and ocean are modeled by the primitive equations of geophysical flows in a hydrostatic formulation. These are conservation laws for mass, momentum, energy, and species expressed as partial differential (and integral) equations. They are time dependent with the ocean surface, forming a boundary condition across which fluxes are exchanged between atmosphere and ocean. Similarly, the land surface with vegetation modeling and soil hydrology and river routing is also coupled with the atmosphere. The physics of the Community Atmosphere Model (CAM) embodies radiation balance with adsorption and emissivity calculated across 16 spectral bands, depending on the chemical constituents of the atmospheric grid point. Moist thermodynamics with cloud water as a prognostic variable and sulfate aerosol dynamics are some of the other formulation specifics of the atmospheric model (Fig. E.2).

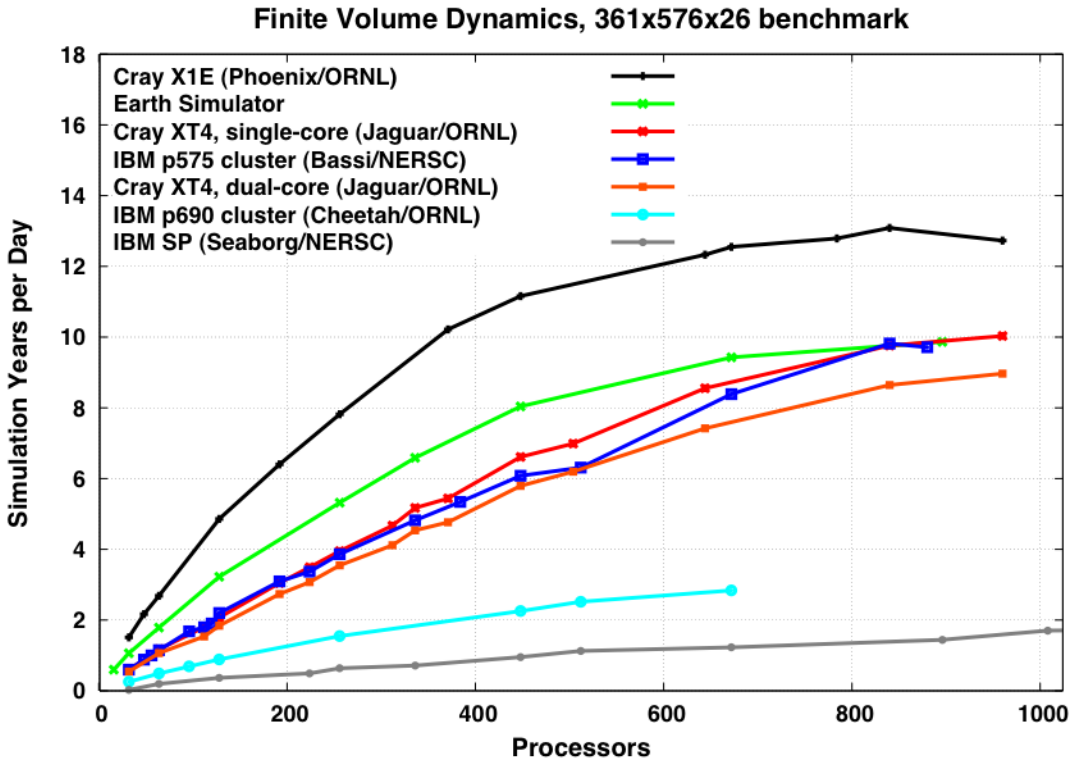


Fig. E.2. Performance of the CAM 3.1 atmospheric model.

Algorithms

The atmospheric model utilizes two different formulations, depending on the form of the discretization. The spectral models (Eulerian and semi-Lagrange spectral) are in "advective form" while the semi-Lagrange finite volume discretization is in "flux form."

Scaling

Scalability has hard limits from the data-distribution algorithm, not from parallel inefficiency. Current development will enable scaling to thousands of processors by increasing resolution, adding computational complexity, and implementing more-scalable data distributions. Two GBs/processor are adequate. MPI with 2-D domain decomposition is the primary mechanism for parallelism. OpenMP parallelism is also implemented and used on systems for which it is appropriate. On the nonvector XT series system, the maximum useful processor counts are higher, though throughput compared with the X1E is lower. The finite volume dynamical core of the atmosphere will also effectively utilize the entire system, particularly with active atmospheric chemistry. In this configuration, the embarrassingly parallel chemistry and physics calculations dominate the dynamics by a factor of 5 to 8.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Fortran 90, C, Co-Array Fortran (optional), GNU Make	MPI, OpenMP (optional), SHMEM (optional), MPI-2 One-sided (optional)	NetCDF	Getrusage (optional)

Math Libraries

Library	Function	Functionality
Cray SciLib	dz/zdfftm (optional)	Multiple real-to-complex or complex-to-real fast Fourier transforms
SGI SCSL	dz/zdfftm1du (optional)	Multiple real-to-complex or complex-to-real fast Fourier transforms

Other Requirements

Good connectivity to the Earth System grid is required.

Code Reference

Mariana Vertenstein (mvertens@ucar.edu)

http://www.ccsm.ucar.edu/models/atm-cam/

NCCS Point of Contact

James B. White III

trey@ornl.gov

CASINO

Physics Models

This code performs a first-principles electronic structure calculation using Quantum Monte Carlo (QMC) to directly solve the Schrodinger equation. In contrast to other first-principles methods, such as density functional theory (DFT), QMC provides essentially exact answers, with no or few approximations in the entire method. The method is therefore ideal for providing benchmark answers for delicate problems such as those in optical properties of nanostructures, catalysis, reaction pathways, and many other problems involving transition metals where common DFT approaches are suspect. Indeed, practical implementations of DFT are based on a parameterization of QMC data. Although calculations are substantially more expensive than DFT, structures of several hundred atoms have been examined. Several QMC algorithms exist; the most accurate involve a set of interacting "walkers" of sets of electron positions that are guided through space by Monte Carlo. Walkers are created and destroyed dynamically according to the underlying quantum problem. The computational requirements scale with the second to fourth power of the number of electrons and atoms, depending on the quantities being measured. A trial wave function partially based on results from a more approximate method (such as DFT) is used to provide importance sampling.

Algorithms

Atomistic QMC calculations have many features in common with both molecular dynamics calculations (e.g., the movement of individual particles, Ewald sums for long range forces) and with quantum chemical and DFT electronic structure methods (e.g., representation of wave functions in an underlying Gaussian or plane-wave basis, possible use of pseudopotentials). A generalized Metropolis algorithm is used for Monte Carlo. The population of walkers is dynamically load balanced across processors ensuring very high parallel efficiency (>90%). The Monte Carlo and dynamic nature of the algorithms could take advantage of fault-tolerant parallel environments, if available: the loss of a few walkers due a failed processor can be rigorously accounted for with only minor overhead.

Scaling

The scalability of QMC calculations depends on a combination of the size of materials system under study, the physical quantities of interest (energies, forces, optical excitations), as well as the quality of trial wave function that can be obtained using more approximate methods. Based on current experience with these governing factors, publication-quality QMC calculations will scale to systems of 1000–10000 electrons on 10000–100000 processors without major developments to existing code. These system sizes are necessary to tackle the problems mentioned above. Hard scaling could be further improved by dividing each walker over several processors. Although this development has not been done, an additional

order of magnitude of scalability might be reasonably achieved. QMC calculations of the type implemented by CASINO have been routinely run on 1000 processors on LLNL systems with >90% parallel efficiency for systems of a few hundred electrons.

If Chosen for Acceptance

The CASINO QMC code has few external dependencies and could be part of an acceptance test. The code stresses the F90 compiler for performance. MPI and communications are not heavily stressed because of the loosely coupled nature of the algorithm.

If Chosen for Science Day One

If chosen for science on day one, it would immediately be possible to study a key scientific problem in an area of materials science such as catalysis, hydrogen production (photodissociation of water on titanium dioxide surface), hydrogen storage in organic and solid state nanostructures, as well as magnetic systems. Calculations on intermediate sized problems (on 10000 cores) are required to determine the exact science and system size that could be achieved on 100000 cores.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O Libraries and functions	Operating System functions
Fortran 90	MPI	None	Timing only

Math Libraries

Library	Function	Functionality
BLAS	xGEMM, xGEMV	Matrix-multiplication Minor use only

Code Reference

Paul Kent (kentpr@ornl.gov)

http://www.tcm.phy.cam.ac.uk/~mdt26/casino2.html

NCCS Point of Contact

Markus Eisenbach

eisenbachm@ornl.gov

CHIMERA

Physics Models

VH-1 solves the compressible Euler equations for fluid flow. It has been coupled to a variety of neutrino transport solvers as part of the Terascale Supernova Initiative. CHIMERA is one of these "code-chimeras," being a combination of VH-1 and the neutrino transport code MGFLD-TRANS (Bruenn, Florida Atlantic U.). CHIMERA solves the equations of radiation hydrodynamics in a ray-by-ray approach: the hydrodynamic evolution is followed in two or three spatial dimensions and the neutrino radiation transport is constrained along radial rays. This is an excellent approximation for the core-collapse supernova problem: for much of the evolution, the configuration is roughly spherical on scales probed by the neutrino interactions with the surrounding matter (Table E.2).

Algorithms

Piecewise Parabolic Method (PPM) is a finite-volume discretization of the Euler equations (a particular example of a Godunov method). VH1 is a Lagrangian remap version of PPM (i.e., the hydro step is performed on a Lagrangian mesh and remapped back to the primary Eulerian mesh during each timestep. CHIMERA includes all the PPM technology of VH-1 along with a fully implicit, multigroup flux-limited diffusion neutrino transport solver. The transport solver uses a variety of Krylov solvers.

Scaling

Explicit Eulerian hydrodynamics is shown to scale to thousands of processors on the NCCS XT series in Fig. E.3. CHIMERA is under active development on the Cray XT series. Its scaling characteristics are essentially identical to VH-1, as the transport solves that mark the added physics in CHIMERA are local.

If Chosen for Acceptance

CHIMERA is under active development and as such is not available to be used as production-level code in an acceptance test.

If Chosen for Science Day One

CHIMERA would perform the world's first core-collapse supernova simulation in 3-D with realistic neutrino transport. The simulation would also likely include magnetic fields, some approximation to general relativistic gravity, and realistic nuclear burning (Table E.2 gives details of calculation for various codes).

A second variant of CHIMERA, bCHIMERA, is also under development. This variant replaces the MGFLD transport in CHIMERA (mCHIMERA) with full Boltzmann neutrino transport.

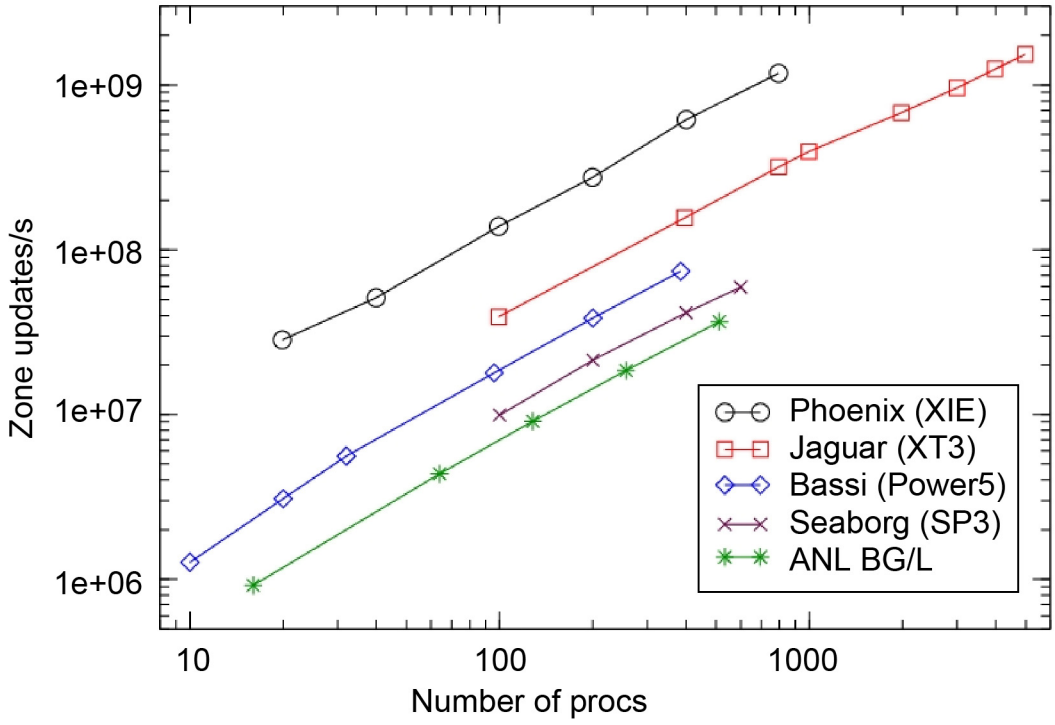


Fig. E.3. Explicit Eulerian hydrodynamics. VH-1 weak scaling.

Table E.2. Details of calculation(s)

Code	Simulation target	Spatial resolution	Phase space resolution	Global memory required	Memory/ process required	Run-time (hours)	MPI processes	Runs
mCHIMERA	Explosion (750 ms)	256 × 128 × 256	20	10 TB	0.6GB	300	16K	3
bCHIMERA	Explosion (750 ms)	128 × 128 × 256	20×8	32 TB	2GB	700	16K	1
ZENITH	Explosion (750 ms)	256 × 128 × 256	20	10 TB	0.6GB	300	16K	3
V3D	Explosion (750 ms)	256 × 256 × 256	20	10 TB	500MB	500	20K	2
GENASIS	Explosion (750 ms)	128 × 64 × 128	20 × 4 × 4	160 TB	8GB	700	20K	1

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
F90	MPI	HDF5, pnetCDF	None

Code Reference

Anthony Mezzacappa (mezzacappaa@ornl.gov)

S. W. Bruenn et al., "Modeling Core Collapse Supernovae in 2 and 3 Dimensions with Spectral Neutrino Transport," *Journal of Physics: Conference Series* **46**, 393–402 (2006).

NCCS Point of Contact

Bronson Messer

bronson@ornl.gov

FLASH

Physics Models

FLASH is designed to solve compressible, reactive flow problems in dense stellar environments, like those found in novae, X-ray bursts, and Type Ia supernova. The code incorporates solvers for hydrodynamics, nuclear burning, gravity, and a variety of other physical processes. The code also has considerable functionality for cosmology problems in the form of particle-mesh solvers.

Algorithms

FLASH uses an explicit, PPM-based method, hence a finite volume, nearest-neighbor code. It uses block-structured AMR. FLASH includes modules to perform passive and active particle tracing, nuclear burning, multigrid and multipole gravity solves, complex equations of state, and front tracking via massive scalar advection.

Scaling

FLASH recently completed a 64,000 processor-driven turbulence run on the LLNL BG/L. The code exhibited good scaling (Fig. E.4).

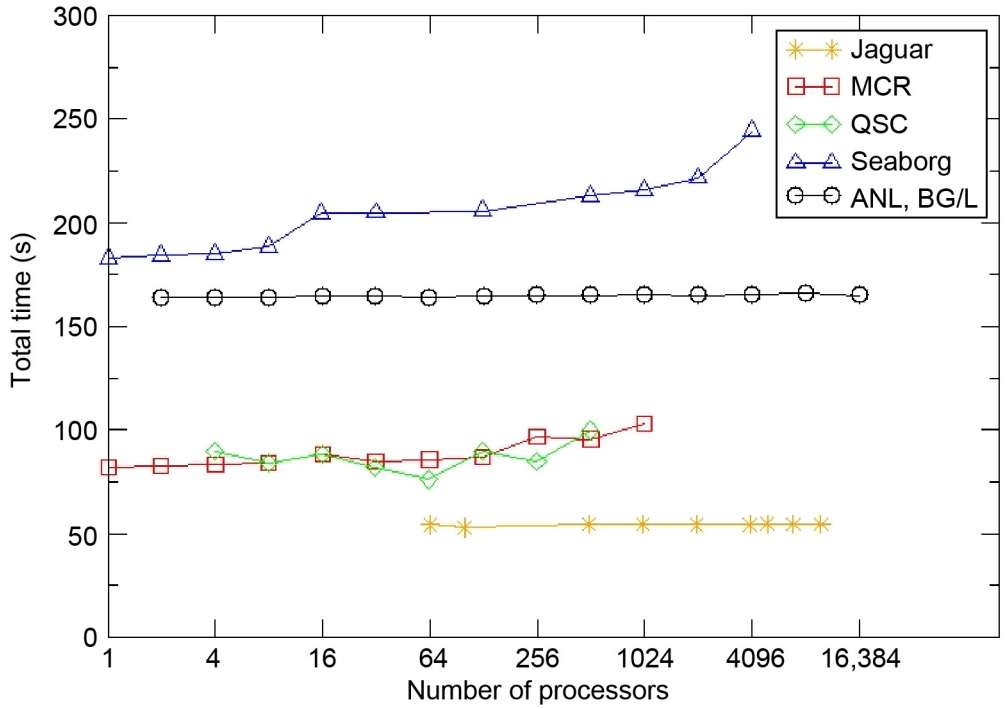


Fig. E.4. FLASH exhibited good scaling.

If Chosen for Acceptance

FLASH could easily be used for a performance (i.e., scaling) acceptance test. Previous experience on BG/L would provide adequate guidance. FLASH would be able to perform turbulence simulation with some prescribed performance boost.

If Chosen for Science Day One

The code could perform a full-star deflagration simulation, including any possible transition to detonation, in the white dwarf at resolutions finer than 0.01 km. This would be a 100× leap in resolution for these kinds of simulations and would allow for real validation of the chosen subgrid model for flame turbulence.

Additional development effort in formulating a new subgrid model would be necessary, along with the development of fast nuclear burning modules designed to capture any deflagration to detonation transition.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Python, F90, C	MPI	HDF5, pnetCDF	GNU make

Libraries and Tools

Math Libraries: The basic FLASH code-base has no external dependencies on math libraries, but users can easily add functionality that changes this.

Code Reference

Don Lamb (lamb@oddjob.uchicago.edu)

http://flash.uchicago.edu

NCCS Point of Contact

Bronson Messer

bronson@ornl.gov

GTC

Physics Models

There are three versions of GTC. GTC, developed at Princeton Plasma Physics Laboratory (PPPL), is a global code for turbulence transport simulations. It uses a shaped plasma in general geometry with electrostatic electron dynamics based on the delta-h scheme with nonadiabatic part of delta-f. The GTC version developed at the University of California–Irvine (UCI) has electromagnetic electron dynamics based on the hybrid scheme along with a global code for both turbulence and gyrokinetic MHD simulations. Finally, the GTC-neo (PPPL) code has neoclassical transport simulations in general toroidal geometry and in fully operational collision operators.

The GTC code has shown steady state simulations of ion temperature gradient (ITG) turbulence with adiabatic electrons. The GTC code developers were able to add the velocity space nonlinearity term, which helps produce an ion current ration of 2.5%. Using ITG simulations with GTC, they were able to show turbulence spreading for shaped and circle plasmas.

ALGORITHMS

Gyrokinetic Vlasov equation PDE in Eulerian coordinates: MHD equations are time dependent PDEs in Eulerian coordinates, and the Gyrokinetic-Darwin-Maxwell equations are time independent PDE in Eulerian coordinates.

GTC solves the Gyrokinetic Vlasov equation using a PIC method (ODE in Lagrangian coordinates). It also solves the Gyrokinetic-Darwin-Maxwell equations with finite elements with multigrid and other linear solvers.

SCALING

Mature PIC code, nearest-neighbor, good scaling to 5000 processors has been demonstrated on a number of systems utilizing MPI and OpenMPI (Fig. E.5). The code has run long simulations on the Cray XT series with 4,800 processors for over 100 wall clock hours per simulation. The code has scaled on over 16K processors on the IBM Blue Gene. The code has shown 96–98% on multicore Opteron processors. GTC has achieved 3.7 TF on the Earth Simulator.

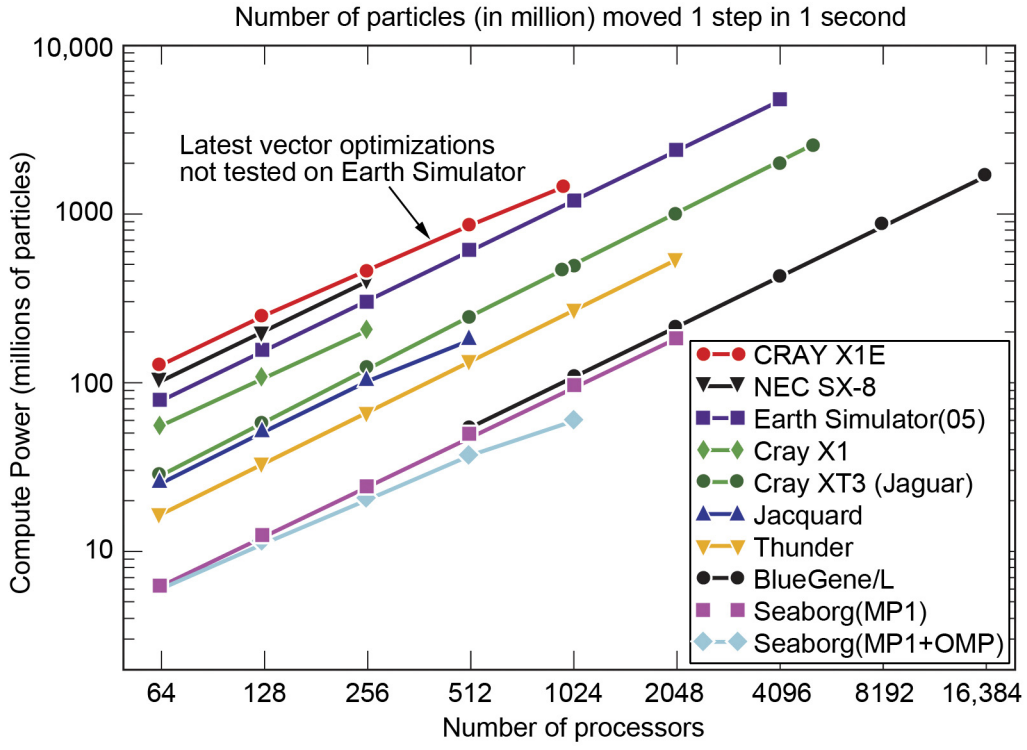


Fig. E.5. Good scaling was achieved on up to 5000 processors. Compute power of the gyrokinetic toroidal code.

If Chosen for Acceptance

- For a 100-TF machine, GTC has shown that it can run today's problems on 16K processors. They will need 36 TB of memory to perform this test, with roughly 2 GB/core. The simulations should run for about 10 h on 16K processors. The basic test will be to see how many particles can be moved one step in 1 s. Currently, the highest has been 6 billion particles on the Earth Simulator with 4K processors.
- For the 250-TF machine, we assume that the 250-TF machine will be available in 2008, and we should be able to use the 2-D domain decomposition (DD) for electrostatic simulation of an ITER-size machine (a/rho>1000) with kinetic electron. So the scaling of 2-D DD will be tested on the 100-TF machine with 20K cores.
- Further down the road, a multispecies, electromagnetic simulation of ITER machine should be carried out on the 1-PF machine (assuming it will be available in 2010), so the finite element method (FEM) solver via PETSc and hypre should also be tested.
- We have used 9 TB RAM on the 25-TF machine already, so it could be assumed that there would be 20 TB on the 100-TF machine. We need at least 2 GB/core.
- The restart file size could be roughly estimated as one-tenth of the RAM size. You could then figure out how much time it would take to write out one-tenth of the RAM (if the bandwidth is known). We would need to write a restart for every 1 h of simulation.
- Write out restarts with <1% overhead on the calculation, writing out about one-tenth of the memory (the particle information).
- Push 20 billion particles 1 step in 1 s, for the 250-TF machines.

If Chosen for Science Day One

For 250-TF: collisionless trapped electron mode (CTEM) physics and transport (heat, particle, and momentum), collisional effects, and size scaling up to ITER; turbulence spreading in ITG, ETG, and CTEM.

- Size and isotope scaling studies of core turbulence transport for ITER: The ultimate goal is for integrated simulation, combining wave heating, turbulence, MHD, and neoclassical physics.
- For 1 PF: Electromagnetic turbulence, long time scale simulation (including transport barriers).
- Understand anomalous particle transport for the electrons in the presence of electromagnetic effects due to micro-tearing near the rational surfaces.

Turbulent transport studies for the energy transport on the 250-TF machines can be carried out using the present GTC code, which uses a grid based on the size of ion gyroradius. The electron particle transport physics, however, requires the incorporation of the electron skin depth in the code, which can be

an order of magnitude smaller than the size of ion gyroradius. Therefore, we need a 100-fold increase in terms of the grid as well as the number of particles. In addition, we need to decrease the time step by a factor of 10 to satisfy the Courant condition due to the smaller grid size.

For the petaflops computer, we will need PETSC fully working with GTC without dominating the over cost of the calculations. This is especially true when solving the field equation, including the electron skin depth.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
F90, C	MPI, OMP (optional)	PHDF5, HDF5, MPI-IO, NetCDF, XML	Timers (through MPI)

Math Libraries

Library	Function	Functionality
PETSC	Hypre	AMG Solver

Code Reference

Zhihong Lin (zhihongl@uci.edu)

S. Ethier, W. M. Tang, and Z. Lin, "Gyrokinetic Particle-in-Cell Simulations of Plasma Micro-Turbulence on Advanced Computing Platforms," *Journal of Physics: Conference Series* **16**, 1 (2005).

NCCS Point of Contact

Scott Klasky

klasky@ornl.gov

GYRO

Physics Models

It is generally accepted within the magnetic fusion community that the dominant cause of cross-field transport in tokamak discharges is plasma microturbulence, and that this turbulence can be described by a combination of the gyrokinetic equation to describe the state of the plasma, and Maxwell˙s equations, to describe the self-consistent electric and magnetic fields. The model equations are often referred to as the gyrokinetic-Maxwell equations.

GYRO is a nonlinear tokamak microturbulence package designed to run on nearly all modern computing platforms, from an ultraportable laptop to the world's largest CRAY X1E/XT series and IBM Blue Gene systems. Developed at General Atomics (starting in 1999) by J. Candy and R. Waltz, GYRO uses a fixed (Eulerian) grid to solve the 5-D gyrokinetic-Maxwell equations. Operation is flexible, with the capability to treat a local (flux-tube) or global radial domain (with an adaptive source to maintain the equilibrium profiles), a full or partial torus, general (Miller shaped) or simple circular plasmas, adiabatic, drift-kinetic or gyrokinetic electrons, electrostatic or electromagnetic fluctuations, finite parallel velocity and shear, and experimental or user-defined physical input parameters. All transport channels are treated: ion and electron energy transport plus turbulent energy exchange, plasma and impurity particle transport, and toroidal angular momentum transport. GYRO is also bundled with a highly-developed GUI-driven IDL analysis package (VUGYRO). Comprehensive code documentation (including a technical manual and user guide), as well as all publications, are available from the GYRO website: http://fusion.gat.com/theory/Gyro. Registered users can download periodic code releases, or work directly from the CVS repository for the most up-to-date version. A GYRO-based SciDAC SAP project (SSGKT) is in progress to develop a steady-state gyrokinetic transport code for predicting reactor plasma profiles given the H-mode pedestal height.

Algorithms

GYRO uses a mixture of finite-difference, finite-element, spectral and pseudo-spectral discretization schemes. Radial derivatives are computed using arbitrary-order finite-difference formulae, whereas 2-D gyroaverages are treated using a mixed spectral (in the binormal direction), pseudo-spectral (in the radial direction). Orbit motion (advection) in the poloidal plane is treated using a third-order upwind scheme, whereas the poloidal field dependence is represented using adjustable-order finite elements. Velocity space integrals (2-D) are computed using novel high-order 2-D Gaussian quadrature schemes, which is the most accurate integration scheme used by any gyrokinetic code (Eulerian or PIC). Time integration is accomplished by either a semi-implicit IMEX-RK scheme (ideal for large, global-scale simulations), or an explicit 4th-order RK scheme (ideal for simulations which resolve the full electron-temperature-gradient physics time and space scales).

Scaling

GYRO scaling studies show impressive scaling up to the full capacity of various of the world's most powerful computers, including 16384 codes of the ORNL Cray XT series (see Fig. E.6). There is active

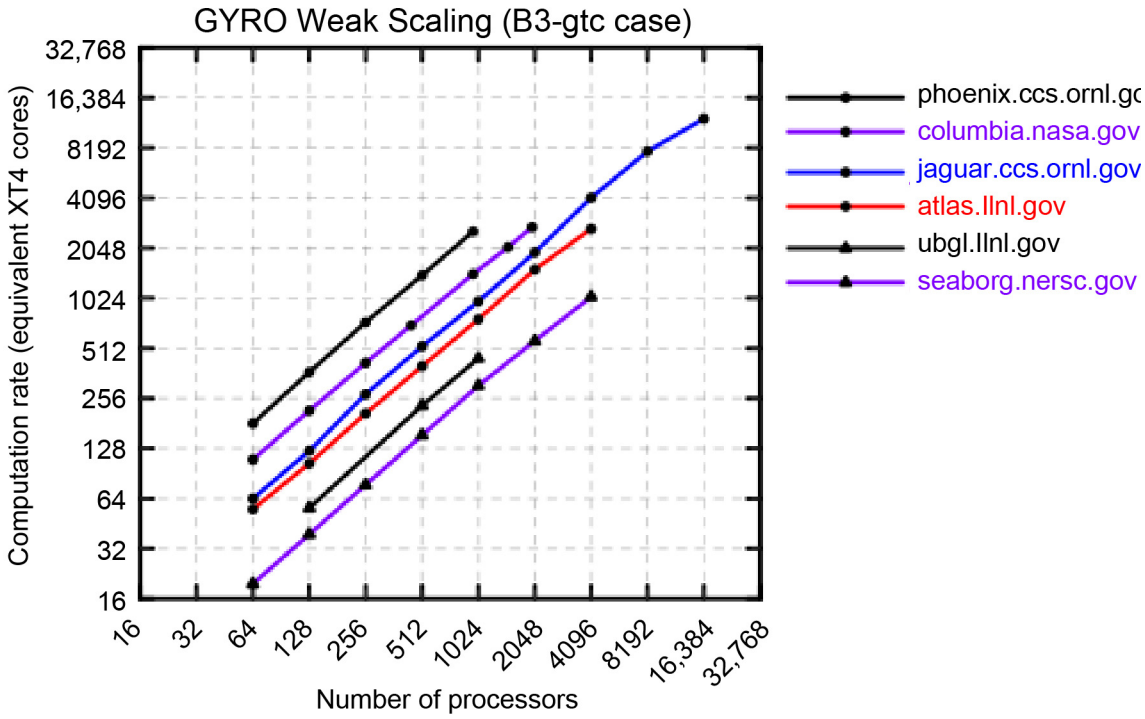


Fig. E.6. GYRO scaling studies on various computers.

development with a SciDAC project to couple multiple instances of GYRO with a single transport module. When complete, the new code (currently named TGYRO) will provide a further dramatic enhancement of GYRO scalability and allow for the efficient use of tens of thousands of cores.

If Chosen for Acceptance

The GYRO code has several external math library dependencies as well as dependence on MPI. GYRO is known to stress the interconnect bandwidth and thus could be used to test that characteristic. Also, GYRO could be used to test parts of FFTW, UMFPACK, and MUMPS.

If Chosen for Science Day One

There are several challenges faced by gyrokinetic simulations. GYRO users have identified these in their attempts to accurately simulate tokamak discharges.

High-beta turbulence: As the plasma pressure (beta) approaches the MHD critical beta, large resonant structures are observed in the electron heat transport. These give rise to large bursty transport and eventually terminate the simulation. Higher-resolution simulations are required to look for clear evidence of destruction of magnetic surface by electromagnetic turbulence.

Pedestal simulations: As the plasma gradients steepen near the plasma edge, recent GYRO results show that grid resolution requirements increase significantly beyond nominal values required to carry out

typical core simulations. Probing the high-transport pedestal region and carrying out studies to determine optional grid resolution in this region are crucial requirements for understanding the transition between core and edge turbulence.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Fortran 77/90	MPI	MPIIO	Timing only

Math Libraries

Library	Function	Functionality
BLAS	ZGEMM, ZGEMV (called from UMFPACK)	Matrix-multiplication Minor use only
LAPACK	ZGETR[F,S,I]	Dense matrix factorization, solve, inversion
UMFPACK	UMZ2FA, UMZ2SO, UMZ21I	Factor, solve, and initialize double complex sparse matrix
MUMPS	ZMUMPS	Driver to initialize, factor, and solve double complex sparse matrix
FFTW or vendor FFT	fftw_f77, fftw_f77_create_plan	Create an object with information required to compute FFT in the FFTW library

Code Reference

Jeff Candy (candy@fusion.gat.com)

http://fusion.gat.com/theory/Gyro

NCCS Point of Contact

Mark Fahey

faheymr@ornl.gov

LAMMPS

Physics Models

Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is a classical molecular dynamics (MD) code developed primarily at Sandia National Laboratories over the past ten years. LAMMPS uses atomistic-based modeling of molecular systems such as biomolecules, material surfaces, and chemical systems. The atomistic modeling uses Newtonian (classical) mechanics for the system

where the atoms are represented by a point mass and charge. Additional terms in the physical model include two-, three-, and four-body terms and pairwise interaction (electrostatic and van der Waals interactions) beyond the fourth body interaction. Computationally, MD is similar to the N-body problem. Unlike gravitational or plasma simulations, the forces in MD are mostly short-range, and particle densities do not reach high values. The timestep in an MD simulation is limited by the need to accurately integrate atomic motion between strongly interacting atoms (e.g., between two atoms coupled by a harmonic bond). For computational efficiency, LAMMPS uses neighbor lists to keep track of nearby particles. The lists are optimized for systems with particles that are repulsive at short distances, so that the local density of particles never becomes too large. On parallel machines, LAMMPS uses spatial-decomposition techniques to partition the simulation domain into small 3-D subdomains, one of which is assigned to each processor. Processors communicate and store "ghost" atom information for atoms that border their subdomain. LAMMPS is most efficient (in a parallel sense) for systems whose particles fill a 3-D rectangular box with roughly uniform density.

Algorithms

A spatial decomposition algorithm and a particle-particle/particle Mesh (PPPM) method and particle mesh Ewald algorithm. Complex 2-D and 3-D parallel FFT are also used.

Scaling

LAMMPS is a highly scalable program, as it has been shown to scale on 64K processors (Livermore BlueGene/L) in weak scaling mode. Scaling and parallel efficiencies are extremely high on the ORNL Jaguar system on the largest allocation (4096 processors). In the replica exchange mode, these simulations can scale to over 100K cores, reducing the time to solution and accuracy of samples significantly.

Typically, the breakdown of CPU cost for a timestep is 85% for force computation, 10% for neighbor finding, and 5% includes time integration, application of boundary conditions, etc. The force computation is dominated by short-range pairwise interactions. Long-range Coulomb interactions are split into a short-range direct portion (van der Waals) and a long-range K-space portion, which is computed by Ewald summation. The most efficient methods for this summation are solutions to Poisson's equation via 3-D FFTs on a grid to which particle charge is interpolated. Ignoring the O(Nlog$N$) cost of FFTs (which typically only require 20–30% of the force computation time), classical MD simulations scale as O(N) in both memory and CPU cost, where N is the number of particles simulated. They also parallelize efficiently, at least for large problems, with typical parallel efficiencies of 80–90% on thousands of processors for simulations with millions of atoms (Fig. E.7).

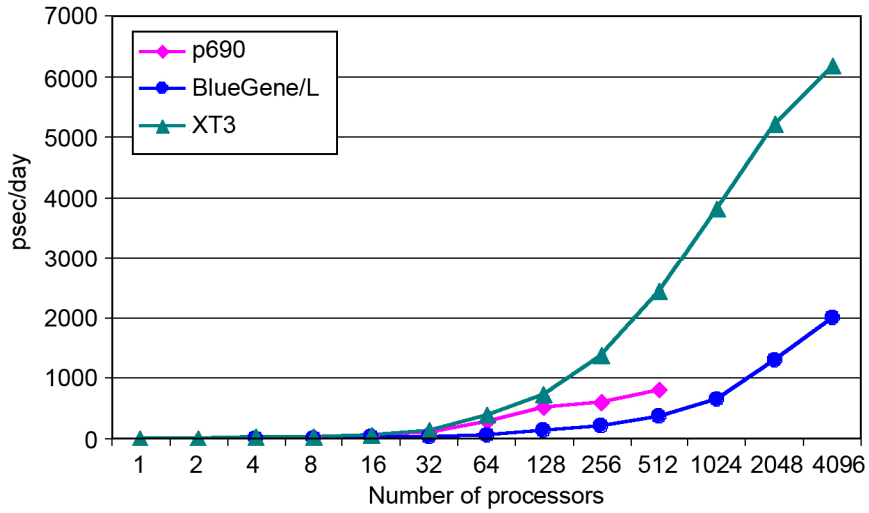


Fig. E.7. LAMMPS parallelize efficiently for large problems.

If Chosen for Acceptance

LAMMPS is compatible with the popular biomolecular force-fields including CHARMM and AMBER. It can perform energy minimization and time integration (molecular dynamics) simulations. Other functionalities include periodic boundary conditions, SHAKE bond and angle constraints, parallel tempering (replica exchange), and targeted molecular dynamics constraints. For all-atom models of proteins or polymers, this requires a time step of about a femtosecond (for coarse-grain models, it can be a few orders of magnitude larger). The current state-of-the-art for supercomputer-scale simulations is that tens of nanoseconds (tens of millions of time steps) can be simulated for models with tens to hundreds of thousands of atoms. This requires many hours or days of CPU time on hundreds of processors of a parallel machine. Similarly, for solid state systems, tens of millions of atoms can be simulated for shorter timescales. Note that this still implies a significant length-scale limitation because there are a few billion atoms in a cubic micron of solid material. Because of their computational intensity, such problems are good stress tests of the performance and scalability of large parallel machines.

If Chosen for Science Day One

LAMMPs runs will allow investigation of complex biomolecular systems with up to 1 million atoms at close to the native time-scales. At the petascale, the scalability of LAMMPS will allow microsecond-millisecond simulations of multimillion atom systems. Data from multiple trajectories (200 to 2000) at microseconds-scale simulations will be collected using the long- and short-range force calculation methods in LAMMPS and by exploiting the replica exchange techniques that could scale to hundreds of thousands of processor cores. These replica exchange simulations improve convergence rates and sampling efficiencies in explicit solvent methods.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
C++	MPI	None	None

Math Libraries

Library	Function	Functionality
FFTW	fftw_create_plan fftw fftw_destroy_plan	Creates/destroys an object with information required to compute FFT in the FFTW library

Code Reference

Steve Plimpton (sjplimp@sandia.gov)

http://lammps.sandia.gov/

NCCS Point of Contact

Sadaf Alam

alamsr@ornl.gov

LSMS (+WL)

Physics Models

This code implements a first principles electronic structure calculation based on density functional theory. LSMS stands for locally self-consistent multiple scattering, an order-N method that is well suited to solve all-electron electronic structure problems as they appear in nanostructures—particularly magnetic nanostructures. The method is formulated within the local spin density approximation to density functional theory and solves the single-particle Dirac equation as well as the nonrelativistic Schrödinger equations. The LSMS code was the first to ever run at a sustained teraflop and was the subject of the 1998 Gordon Bell Prize.

Algorithms

LSMS solves the Kohn-Sham equations of density functional theory using Multiple Scattering theory to calculate its Green function and consequently the resulting densities by calculating the trace of the product of the observables and Green's function. The main computational effort involves inverting a matrix of dimension that scales linearly with the size of the system. To achieve linear overall scaling with system size, LSMS takes advantage of the fact that most observables only depend on their local

environment, so by taking only a fixed size neighborhood of atoms into account, LSMS keeps the size of the matrices independent of the system size after the range of this local interaction zone has been determined.

Scaling

LSMS has run on 10,000 processors with excellent scaling. Parallelization is achieved by assigning system atoms to different processors. Larger system calculations are enabled by time on LC systems. LSMS has been run on the BG system with either one task per core or one task on two cores. To be efficient for one task on two cores, an implementation of ZGEMM that takes advantage of the multiple cores is needed. LSMS presently does not provide for structural relaxation (this is currently under development), therefore a second code is needed, probably VASP. Integration of these ab initio methods with a classical statistical physics method (generalized Wang-Landau in particular) as the energy function will allow another level of parallelism in the random walkers used. This combined code will naturally scale to >100,000 cores when investigating the thermodynamic behavior of 1000–10,000 atom nanoparticles.

If Chosen for Acceptance

LSMS can be run for a system of any size up to the number of cores available (because of the one-to-one mapping of atoms and processors). A simple well-known bulk system (e.g., bcc Fe) can be used to test the stability and scaling of message-passing performance up to the full machine size.

If Chosen for Science Day One

The combined LSMS+Wang-Landau code will allow the computation of the temperature dependent magnetic free energy for nanoparticles of interest (FePt in particular). The use of a first principles-based method will take into account effects due to chemical order and will study the full range of magnetic complexity in these nanoscale systems. Not much is known about the sub-nanoscale magnetic structure of these particles, and this information is central to understanding and exploiting the magnetic properties of FePt nanoparticles. The achievability of these results will depend on the number of WL samples needed to compute free energies to sufficient accuracy.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Gnu make, Fortran 77/90, C, C++	MPI2	Serial HDF5, XML	ctime (or equivalent)

Math Libraries

Library	Function	Functionality
BLAS	ZGEMM	Dense double complex matrix-matrix multiply
BLAS	ZGEMV	Dense double complex matrix-vector multiply
LAPACK	ZGETRF	Double complex factorization of a dense matrix
LAPACK	ZGETRS	Double complex triangular matrix solve
LAPACK	ZGETRI	Double complex matrix inverse formation

Code Reference

Thomas Schulthess (schulthesstc@ornl.gov)

C. G. Zhou, T. C. Schulthess, and D. P. Landau, "Wang-Landau Algorithm for Continuous Models and Joint Density of States," *Physical Review Letters.* **96**, 120201 (2006).

NCCS Point of Contact

Markus Eisenbach

eisenbachm@ornl.gov

MADNESS

Physics Models

Numerical-based simulations will be used to predict the physical and chemical properties of molecules.

Algorithms

The mathematical, algorithmic and computational techniques used in MADNESS are based upon these elements:

- Multi-resolution analysis in multi-wavelet bases
- Separated representations of functions and operators
- Partitioned singular value representations
- Bandwidth-limited bases for efficient sampling in space and evolution in time

Scaling

Although MADNESS is under continuous development, the current scaling on the Cray XT series system at ORNL shows good overall scaling and scalability of the component algorithms (Fig. E.8).

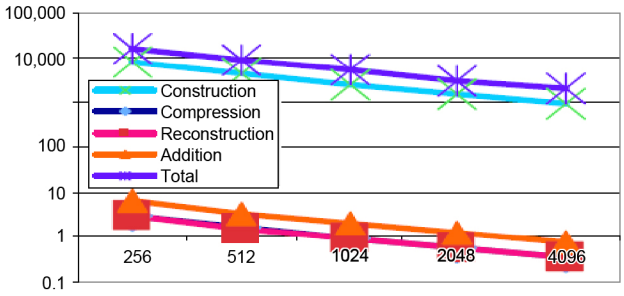


Fig. E.8. MADNESS shows good overall scaling and scalability of the component algorithms.

If Chosen for Science Day One

We propose to develop and apply a petascale simulation capability for essentially exact simulation of the dynamics of a fully interacting few-electron system (He, H2, H3+, Li, LiH) in a general external field (i.e., propagation in 6-D over physical time scales of the wave function of few-electron atoms or molecules in the presence of a perturbing particle such as photon, electron, proton or antiproton). These are the fundamental and defining challenges in physics and chemistry of the 21st century, for which scientists have been seeking solution for more than 50 years. Passing this frontier will open completely new areas to quantitative scientific inquiry and pave a path to similar capabilities for many-electron systems because the many-electron wave function is very well approximated by (non-)linear combinations of pair functions. With current computing resources and numerical techniques, this is presently impossible, and when the equivalent task was recently first accomplished for one+one-electron systems in 3-D or more, the result was reported by several cover-page articles in *Science* and numerous other publications, including *Nature*. We believe this to be feasible due to our current work in 3-D, prototyping in 6-D for stationary bound states, and the use of petascale computers with vast memory and computational power. The numerical tools and software will be based upon MADNESS. We will generalize this capability to time-dependent, nonstationary states, and develop multiscale schemes to solve and propagate the few-electron Schrödinger equation with high accuracy.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Fortran	MPI		

Math Libraries

Library	Function	Functionality
BLAS		

Other Requirements

The overall communication patterns and algorithms desired for the full development of dynamic algorithms for the MADNESS suite include the ability to utilize at least four complex programming models: active messages, unified parallel C, global arrays, and message passing (MPI). These models not only need to be available but must co-exist and allow for functional interoperability.

Code Reference

Robert J. Harrison (harrisonrj@ornl.gov)

http://code.google.com/p/m-a-d-n-e-s-s/

NCCS Point of Contact

Rebecca J. Hartman-Baker

hartmanbakrj@ornl.gov

MILC/Chroma

We propose a set of closely related projects in the numerical study of quantum chromodynamics (QCD) to be part of the early research program on the ORNL 250-TF Cray XT series. QCD calculations address problems that are at the core of DOE's large experimental programs in high energy and nuclear physics, problems on which the Cray XT series can have a major impact.

Lattice QCD calculations are performed in two steps. In the first, one performs Monte Carlo calculations to generate gauge configurations, which are representative samples of the QCD ground state. These configurations are stored, and, in the second step, they are used to calculate a wide variety of physical quantities. Configuration generation is computationally intensive, but the memory, I/O, and storage requirements are modest. The code is compact and relatively straightforward to optimize. Jobs are run in a small number of streams and can be handled by a few people. By contrast, the calculations of

physical quantities from the configurations typically require many fewer floating-point operations, but have significantly greater I/O and storage needs than configuration generation.

The physics analysis codes are more complex and, in some cases, more difficult to optimize. More researchers are usually involved in the physics analysis work. For these reasons, the projects we propose for early use of the 250-TF Cray XT series are the generation of gauge configurations, the part of our calculations in which this system can quickly produce unique results that will have major impacts on high energy and nuclear physics. Determination of physical quantities from these configurations will initially be done on clusters, the QCDOC, and at national supercomputer centers, but later in the project, we propose to move some of this work to the Cray XT series.

All configurations generated on the Cray XT series will be made available immediately to all members of the US lattice QCD community for use in a wide range of physics applications.

One major scientific goal of the physics community is to determine the effects of the strong interactions (QCD) on weak interaction processes to an accuracy needed to make precise tests of the Standard Model, our current set of theories describing fundamental interactions. These calculations are critical for a number of major ongoing experiments in high energy physics, including BaBar at the Stanford Linear Accelerator Center (SLAC), CDF and D0 at the Fermi National Accelerator Laboratory (FNAL) and CLEO-c (Cornell University). A second major goal is to calculate the masses of the strongly interacting particles and obtain a quantitative understanding of their internal structure and interactions.

This work is very important for major nuclear physics experiments including RHIC at Brookhaven National Laboratory (BNL) and CEBAF at Jefferson Lab (JLab). To obtain accurate numerical results, one must generate gauge configurations with a range of lattice spacings in order to extrapolate to the continuum (zero lattice spacing) limit, and for a range of light quark masses to extrapolate to their physical value. Configurations with the smallest lattice spacings and lightest quark masses anchor these extrapolations, and ultimately determine their accuracy. To obtain the level of accuracy needed to fully support the experimental programs in high energy and nuclear physics, it is necessary to generate gauge configurations with smaller lattice spacings and lighter up-and-down quark masses than has been possible up to now.

The Cray XT series planned for ORNL will enable us to generate such configurations. We expect these configurations to resolve long-standing problems, and substantially improve calculations of the mass spectrum of strongly interacting particles, our understanding of the structure and interactions of nucleons, and the extraction of fundamental parameters of the Standard Model from experimentally measured weak matrix elements.

Physics Models

In order to carry out numerical studies of QCD, it is necessary to formulate the theory on 4-D space-time lattices. During the past few years, a great deal of progress has been made through the use of improved formulations of lattice QCD (improved actions). The USQCD Collaboration, which consists of nearly all the lattice gauge theorists in the United States, is making use of the three formulations we consider to be the most promising: the improved staggered (Asqtad) action, the domain wall fermion (DWF) action, and the Wilson-Clover action. Each of these actions has important strengths for addressing different physics questions: The Asqtad action is computationally efficient, and is enabling precise tests of the Standard Model; the DWF action possesses nearly exact chiral symmetry for finite lattice spacing, eliminating many problems associated with operator mixing; and the anisotropic Wilson-Clover action enables correlation functions to be examined at short distances to extract excited states. Furthermore, it is essential that we validate our results by calculating some quantities with more than one of these actions. For these reasons, we propose to generate gauge configurations on the ORNL Cray XT series with all three actions. We describe each of these actions below, and, in Table E.3, we set out projects that we believe are appropriate to the ORNL 250- and 1000-TF machines.

Table E.3. Proposed gauge configurations

Lattice spacing (Fermi)	m_l/m_s	Lattice dimensions	Monte Carlo steps	Cray (peak TF)	TF years
Asqtad gauge configurations					
0.045	0.40	$56^3 \times 192$	4000	250	0.6
0.045	0.20	$56^3 \times 192$	5000	250	1.9
0.045	0.10	$80^3 \times 192$	6000	250/1000	13.7
0.060	0.05	$84^3 \times 144$	5000	1000	18.4
DWF gauge configurations					
0.094	0.27	$32^3 \times 64$	4500	250	1.2
0.094	0.19	$48^3 \times 64$	5000	250/1000	7.8
0.094	0.11	$48^3 \times 64$	9000	1000	22.6
Wilson-Clover gauge configurations					
0.10	0.22	$32^3 \times 128$	50000	250	0.8
0.10	0.15	$40^3 \times 128$	60000	1000	4.1
0.08	0.18	$40^3 \times 128$	40000	250/1000	4.5
0.08	0.15	$48^3 \times 128$	50000	1000	22.0

Asqtad gauge configurations: The Asqtad action has the advantage that it requires an order of magnitude fewer floating point operations to generate gauge configurations with a particular lattice spacing and light quark mass than other improved actions. For this reason, a large set of gauge configurations ensembles already exists. These configurations have been made publicly available and are being used by many lattice gauge theorists in the United States to study a wide variety of problems in high energy and nuclear physics. Accuracy in the order of 3% has been obtained for a select set of physical observables, and, in some cases, predictions have been made that were later confirmed by experiment. However, to reach the level of accuracy for precise tests of the Standard Model requires gauge configurations with lighter quark masses and smaller lattices spacings than have been possible to generate to date. The Cray XT series will enable us to generate such configurations, which will have a dramatic impact on a wide range of calculations. The MILC Collaboration, which has generated the Asqtad gauge configurations mentioned above, will have responsibility for this project.

DWF gauge configurations: A critical advantage of the DWF action is that it satisfies nearly exact chiral symmetry for finite lattice spacings. The spontaneous breaking of chiral in the QCD vacuum is the origin of most of the normal matter in the universe, and an accurate representation of this symmetry is ultimately essential for lattice QCD calculations of the weak matrix elements, the spectrum of strongly interacting particles, and the structure and interactions of nucleons, which are central goals of research for the USQCD Collaboration.

At present, the DWF action is an important part of the efforts of the USQCD Collaboration both for calculations on existing gauge ensembles and for the generation of new ensembles. Unfortunately, current efforts to generate Monte Carlo ensembles using these new methods have been statistics starved and constrained to work with a relative coarse lattice spacing of 0.125 fm. Dramatic improvements can be achieved using the Cray XT series, staged according to the expected availability of equipment. By exploring the light quark region with reasonably small lattice spacing, the full promise of the DWF formulation should be realized. The RIKEN/BNL/Columbia University (RBC) and Large Hadron Physics Collider (LHPC) Collaborations, which have led the development and use of the DWF action will have responsibility for this project

Wilson-Clover gauge configurations: A complete understanding of QCD demands that we know the spectrum of mesons and baryons that it implies, and test these spectra against high quality data.

The combined analysis of experimental data on the photo production of nucleon resonances is the nuclear physics 2009 milestone in hadronic physics, and the GlueX Collaboration's proposal to explore the spectrum of exotic mesons is a flagship component of the proposed 12-GeV upgrade at Jefferson Laboratory. Beyond the spectrum of isolated hadrons, lattice QCD can teach us about the mechanism of hadronic interactions, a hadron physics milestone for 2014.

Thus the need for a comprehensive, first-principles study of the spectrum of QCD, and of the nature of hadronic interactions using lattice calculations is clear. These studies require precise computations of several low-lying energy eigenvalues, necessitating good resolution of the temporal decay of correlation functions. These requirements lead us to adopt anisotropic lattices, in which the temporal lattice spacing is finer than the spatial by around a factor of three. The LHPC Collaboration will take responsibility for this project.

Table E.3 shows the gauge configurations we propose to generate on the ORNL 250- and 1000-TF computers. The first column gives the lattice spacing in fermi, the second the ratio of light-to-strange quark mass, the third the lattice dimensions, the fourth the number of Monte Carlo steps in the simulation, the fifth the peak speed of the ORNL computer on which we propose to generate the configurations, and the sixth column the estimated number of floating-point operations required in teraflop years. In the case of the Wilson-Clover gauge generation, we list the minimum value of the light-to-strange quark mass used in each stage of the calculation.

Algorithms

The generation of gauge configurations will be carried out with the recently developed Rational Hybrid Monte Carlo (RHMC) algorithm. This algorithm provides a major improvement over older ones. Indeed, our proposed work could not be accomplished without it. The single most computationally intensive step in our calculations is the inversion of large sparse matrices, which is performed using the conjugate gradient algorithm.

Scaling

Both the Chroma and MILC codes, which will be used in the proposed work, achieve excellent scaling on the Cray XT series. This is demonstrated in Fig. E.9, where we plot the total throughput for the conjugate gradient routine for both codes as a function of the number of cores on the recently upgraded ORNL Cray XT series. This performance, approximately 18% of peak, was obtained with vanilla versions of the two codes. We indicate our plans to optimize the codes for the Cray XT series below.

If Chosen for Acceptance

We nominate the MILC code to be one of the codes used in acceptance tests. It is publicly available at the URL http://www.physics.utah.edu/˜detar/milc, and has been used in recent procurements by National Energy Research Scientific Computing Center (NERSC) and the National Science Foundation (NSF). The Asqtad action has the highest ratio of data movement to floating point operations of the QCD actions in

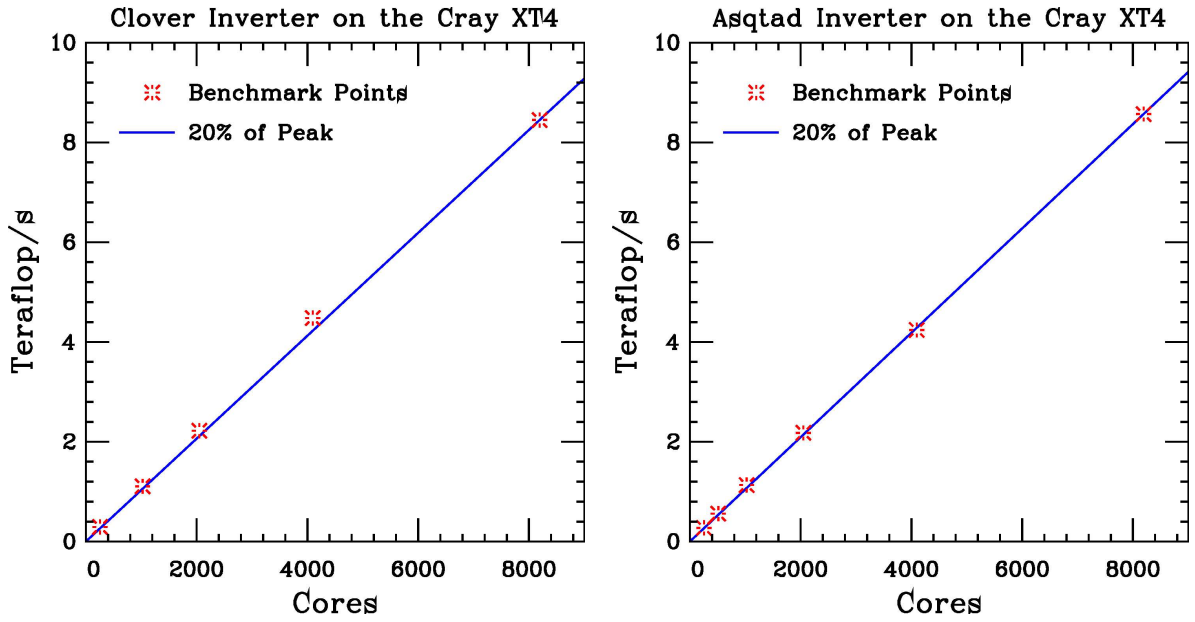


Fig. E.9. The codes for the Cray XTE will be optimized. Performance of the Chroma (left panel) and MILC (right panel) codes for the conjugate gradient routine as a function of the number of cores. The number of lattice points assigned to each core is held fixed as the number of cores is increased. The black curves are straight lines passing through the data from 1024 and 2048 cores.

current use for large-scale simulations. It therefore provides a particularly good test of each of these functions and the balance between them.

The vanilla MILC Code with SciDAC enhancements achieved 18% of peak for the conjugate gradient routine on the current version of Jaguar without special optimization. We suggest that a reasonable acceptance metric would by 15% of peak, again without special optimization.

IF CHOSEN FOR SCIENCE DAY ONE

We would begin by generating the Asqtad configurations with lattice spacing a = 0.045 fm and light-to-strange quark mass ratios of ml/ms = 0.40 and 0.20, the DWF configurations with a = 0.094 fm and ml/ms = 0.27, and the Wilson-Clover configurations with a = 0.10 fm and ml/ms down to 0.22. These projects could be completed quickly on the 250-TF machine, and would have an immediate impact on our field. We would then begin the Asqtad configurations with a = 0.045 fm and ml/ms = 0.10, the DWF configurations with a = 0.094 fm and ml/ms = 0.19, and the Wilson-Clover configurations with a = 0.08 fm and ml/ms = 0.18. Completion of these projects may turn out to be good initial tests for the 1000-TF machine. The Asqtad configurations with a = 0.045 fm would reduce errors in physical quantities due to lattice artifacts by a factor of two over those arising from configurations expected to be available in FY 07. The completion of the DWF configurations with a = 0.094 fm will allow the exploration of the chiral regime with chiral fermions, and enable calculation of fundamental matrix elements that would suffer from operator mixing in the absence of chiral symmetry. The completion of

the Wilson-Clover configurations at a = 0.10 fm will enable the computation of the exotic meson spectrum and of the low-lying excited baryon resonances down to pion masses as low as 180 MeV, and the first measurement of the exotic meson photo couplings. The completion of the lattices at a = 0.08 fm will enable the continuum limit of these quantities to be determined.

The objective of all of these projects is to generate gauge configurations that are representative samples of the QCD ground state. More specifically, importance sampling techniques are used to generate configurations with a probability that is proportional to their weight in the Feynman path integrals that define the theory. These configurations will be saved, and will be used to calculate a wide variety of physical quantities. As indicated above, the RHMC algorithm will be used in all of the proposed work. The bulk of the floating point operations are consumed in inversions of large sparse matrices, which are performed by the conjugate gradient algorithm.

As can be seen from Fig. E.9, both the Chroma and MILC codes currently obtain approximately 18% of peak on Jaguar. They will therefore be ready to run effectively on the 250-TF machine without any further development effort. However, we are confident that the performance of our codes on the Cray XT series can be improved significantly, and we are working to do so. There are two areas that will be given special attention. First, the basic building blocks in QCD calculations are linear algebra operations among 3 × 3 complex matrices and three-component complex vectors. The most frequently used of these have been optimized for Intel processors with SSE instructions. However, this code does not produce the boost in performance on Opteron processors that it does on Intel ones, so we plan to optimize key linear algebra operations for Opterons either using SSE instructions or assembly coding. Second, our codes make use of multi-core processors by treating each core as an independent processor, and running a single MPI process on it. However, we believe that threaded code will provide better performance, and plan to develop it. Both of these optimization efforts are planned for the first year of the lattice gauge theory community's SciDAC-2 grant, and should be completed by the time the 250-TF Cray XT series comes on line. The Lattice QCD Software Committee has responsibility for this work.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
C, C++	MPI	POSIX compliant I/O system calls and large file (>2 GB) support	Standard UNIX like system calls (current QK kernel functions appear sufficient)

Other Requirements

Also required are C/C++ compilers that allow programmer control of SSE and Opteron-optimized instructions, and memory prefetching through compiler intrinsics or inline assembler (e.g., GNU Compiler Collection (gcc/g++) v 3.4 or higher), PGI compilers are insufficient because auto vectorization is not sought. The PathScale compilers have also proved usable. Cray SHMEM is desirable for optimizing our communications libraries, but not essential.

Code Reference

Robert Sugar (sugar@physics.ucsb.edu)

MILC: http://physics.indiana.edu/~sg/milc.html

CHROMA: http://usqcd.jlab.org/usqcd-docs/chroma/

NCCS Point of Contact

Ricky Kendall

kendallra@ornl.gov

NEWTRNX

Physics Models

Neutronics models include a 6-D neutral-particle Boltzmann transport equation for neutron distribution coupled, for time-dependence, with the Bateman equations for isotopic and delayed neutron generation/destruction. The requirements for the Boltzmann solution span six to nine orders of magnitude in space, three to four in neutron direction, and two to four in neutron energy. The Bateman equations for isotopic generation/destruction are solved for every spatial element or region with over 2000 coupled isotopes and spanning 11 orders of magnitude in time (milliseconds to decades). Other phenomena to be considered are coupling with multiphase fluid flow, structural mechanics, fuel behavior, and nuclear chemistry. (Note: this is the neutronics solver and computational backplane of Collaboration for Advanced Nuclear Simulation [CANS]: a multi-institution collaboration with Idaho National Laboratory (INL), Argonne National Laboratory (ANL), Los Alamos National Laboratory (LANL), and several universities to develop high-fidelity coupled-physics simulations for nuclear reactors.)

Current simulations of nuclear reactors are a four-step process to solve the equation and approximate their coupling with fluid-flow and heat transfer:

- Fine-energy (based on first-principles nuclear scattering) and spatial resolution on a very small spatial subset (single fuel particle) to estimate local effects of the global solution and weight the continuous energy cross sections to provide an accurate “smeared continuous-energy section set” for this spatial subset for a given state point (temperature, density, time).

- Fine-energy (based on first-principles nuclear scattering) and spatial resolution on a larger subset of the problem (a single fuel type) to estimate the local effects of the global solution and the weight of the spatially-smeared continuous-energy cross sections to provide an accurate "effective spatially smeared multigroup cross section set" for this fuel type for a given state point.
- Multigroup in energy, with a coarser spatial resolution, on a larger piece of the problem to the weight of the multigroup cross sections to provide an accurate "effective two-group homogenized cross section set" for many perturbations in state-points (i.e., looping through steps one-three) to approximate the effect of various state points and coupling with the Bateman equations.
- Two-group in energy, large homogenized (smeared) material regions for the full spatial domain, with approximate coupling to 1-D thermal-hydraulics solvers.

Each step requires several million degrees of freedom and the major approximations reflect boundaries at each level, the coarse-level temperature feedback approximation, and the fine-mesh reconstruction from the coarse-solution. Many safety issues need to know peak fuel temperature (or similar data) so it must be "reconstructed" from a combination of the local assembly calculation and the global diffusion calculation.

The NEWTRNX code incorporates the world's leading tools in ORNL's SCALE code package for steps one and two while solving the multigroup solution for the entire spatial domain (10^{15} degrees of freedom). The coupling with the Bateman equations and computational fluid-dynamics will be incorporated in FY 09. Also, the fine-energy treatment, which is based on first-principles nuclear scattering, will be available in future releases of NEWTRNX to remove the step two entirely. However, this will not be practical for a full commercial nuclear reactor until computing resources expand by several orders of magnitude (tens of petabytes and hundreds of petaflops).

Algorithms

The Boltzmann PDE is solved with the method of characteristics (slice-balance approach) in space, multigroup in energy, and discrete ordinates in neutron direction. Multigrid algorithms in 6-D phase space will be used for acceleration of the solution along with Krylov solvers. The sweeping algorithm allows for a matrix-free formulation which reduces the dimensionality of the solution vector, allowing reduced storage requirements and efficient solutions for 10^{13} unknowns on thousands of processors in less than an hour.

Scaling

In Fig. E.10, the strong scaling performance of NEWTRNX from 32 to 512 processors of Jaguar is shown for a problem with a single representative neutron energy-group, 80 discrete-ordinate directions,

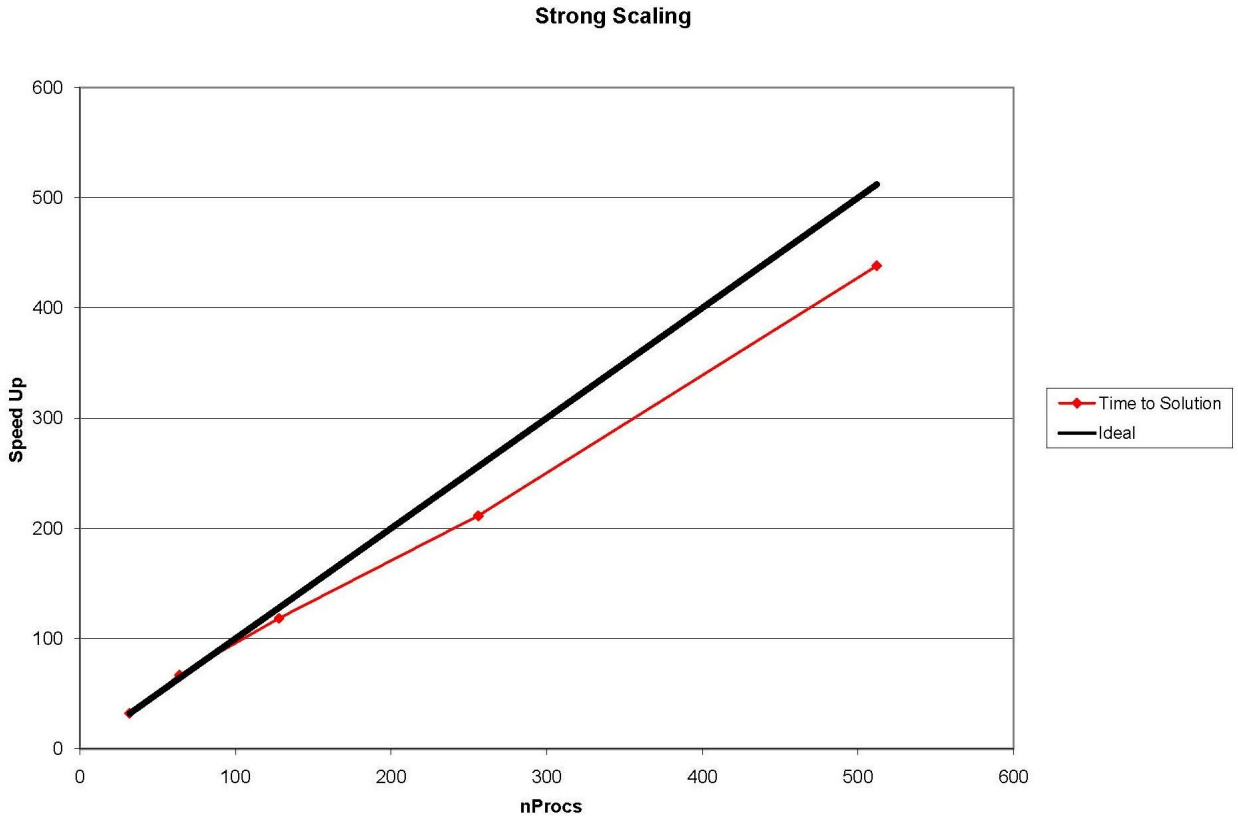


Fig. E.10. Single fuel assembly of a sodium-cooled, fast-spectrum nuclear reactor.

and 524,742 tetrahedral elements (167 million DOF). As with UNIC, the problem selected for this demonstration was chosen to study the performance on Jaguar, but the practical problems studied will require orders of magnitude higher resolution in phase-space, which will increase the local work load and improve parallel performance. Even though the spatial domain goes down by a factor of 16 (from 16k to 1k elements per core), we still see reasonable parallel performance.

This demonstrates that the Parallel-Block Jacobi algorithm, accelerated with a non-linear multi-grid solver, scales well for large problems with distributed sources and weak coupling between dispersed spatial blocks. As shown, the algorithm scales at 85% on 512 processors (with respect to 32 cores) on Jaguar for a problem with spectral radius of 0.65, which is also representative of the performance of this algorithm for reactor configurations.

If Chosen for Acceptance

The demonstration of a high-fidelity (in all phase-space) simulation of the neutron distribution within a nuclear reactor would revolutionize the nuclear industry, providing a functionality that could not be realized for generations with the present single-processor tools of today's nuclear industry.

The ability, never before available, to model an entire nuclear reactor to such fidelity and comparison with empirical data from present international test reactors would prove the significance of high-performance computing to the nuclear community.

If Chosen for Science Day One

Six dimensional (3-D space, 2-D direction, 1-D energy) neutron transport for an entire nuclear reactor core consisting of 250 fuel assemblies, with each assembly holding 250 fuel pins (150 fuel pellets per pin). A total of ~10M fuel pins will be simulated, with the solution being the energy distribution/output throughout the entire reactor core for a given fuel type and age.

The idealized solver would provide fine resolution in every dimension without the current smearing steps, or without the use of fine-energy and spatial resolution on the entire problem with complete temperature feedback. This includes 3-D S_n, near-continuous energy transport on the entire reactor; S_{16} (or higher) quadrature set, 10^{12} spatial elements, and 30,000 discrete energy points; one calculation per temperature feedback iteration (<10) per quasi-static time step (~100) for a basic calculation; or a total of 10^{21} unknowns performed 1000 times. Some reactor types will have a very strong thermal feedback coupling, most will be relatively weak.

Advanced phase-space domain decomposition must be incorporated for efficient scaling to 100,000 processors.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
F90, C, C++, python	MPI, cca-tools	HDF5	GNU make

Math Libraries

Library	Function	Functionality
LAPACK	DGEEV	Compute eigen decomposition
PARPACK	PDNEIGH	Compute eigenvalues and Ritz estimates

Code Reference

Kevin Clarno (clarnokt@ornl.gov)

K. T. Clarno, “Implementation of Generalized Coarse-Mesh Rebalance in NEWTRNX for Acceleration of Parallel Block-Jacobi Transport,” *Transactions of the American Nuclear Society* **97**: 498–500 (2007).

NCCS Point of Contact

Doug Kothe
kothe@ornl.gov

NUCCOR

Physics Models

The nuclear coupled cluster — Oak Ridge (NUCCOR) code was built from scratch. Chemistry codes cannot be adapted to the nuclear problem since the nuclear forces are spin and isospin dependent. Thus, standard algorithmic savings in chemistry brought about by the symmetries of the Hamiltonian (e.g., use of a spin-orbital basis) cannot be employed for nuclei. Coupled-cluster techniques solve for ground- and excited-states of a quantum many-body system at a given level of many-body sophistication. The ground state energy and the cluster amplitudes result from an iterative solution of a large set of nonlinear, coupled algebraic equations. A compact mathematical statement of the problem is given by the expression $f_i(t_1,t_2) = 0$, where the set of unknown amplitudes (one-particle-one-hole and two-particle-two-hole excitation amplitudes t_1 and t_2) must be found. The equations are closed if one assumes that any higher-order amplitudes (three-particle-three-hole amplitudes in this case) are zero. Energies of excited states are computed via diagonalization of a large-dimensional, sparse-eigenvalue problem.

Algorithms

Solution of a nonlinear set of coupled algebraic equations. A complete calculation for a give nucleus proceeds in the following manner. First, one must generate the effective two-body interaction for the problem. This is done by renormalization of bare nucleon-nucleon potentials via sums of ladder diagrams (the G-matrix approach), a Hamiltonian similarity transformation and projection to the model-space, and a renormalization group (RG) method that obtains the low-momentum part of the interaction. This step is not numerically intensive and can be performed on a small cluster or workstation. The RG approach, also known as Vlowk, will be utilized to investigate three-body effects. Second, the two-body interactions obtained from the first step are calculated in a "spin-coupled" representation and must be decoupled. This procedure is performed by first reading in the coupled matrix, and using a master-worker algorithm to spread the work of decoupling. Once matrix elements have been decoupled and MPI-I/O written to a file, the resulting 4-index array of matrix elements is block-distributed among the processors with a MPI-I/O read. This is an extremely efficient (and crucial) part of the overall algorithm. The final step involves calculation of the NUCCOR amplitudes. The present code uniformly distributes the interaction matrix elements across processors on two of the four indices. Each processor maintains a complete copy of the amplitudes. Thus each processor performs a partial sum of the equations to obtain new amplitudes. An

allreduce (addition) is used to obtain the new copies of the amplitudes for the next iteration step. While keeping one copy of the amplitudes for each processor means that as we go to larger model spaces, memory use becomes an issue, the overall flops performance of the code has benefited from this strategy. For non-iterative triples corrections, we transport the resulting NUCCOR amplitudes and decoupled Hamiltonian to collaborators at Michigan State for analysis

The code currently relies upon a significant use of regression tests. If we change codes, we check previous results. We also have pilot codes running serially for small problems which can then be tested in parallel applications. We also have "standard results" produced by other methods to which we can compare. We have diagonalization results to compare to for special cases. Comparison to experimental data for nuclear properties is our validation strategy. In nuclear physics, where the Hamiltonian is less well known, this can be a particularly challenging issue; however, given the same Hamiltonian, completely different methods (GFMC vs coupled-clusters, for example) should obtain the same results. Direct comparison to experiments is becoming more and more common. Data flows from HRIBF at ORNL, Atlas at Argonne National Laboratory (ANL), the National Superconducting Cyclotron Laboratory (NSCL) at Michigan State University (MSU) as well as from international facilities.

SCALING

The computational requirements scale as $N_o^2 N_u^4$, where N_o and N_u are the number of occupied and unoccupied single-particle orbitals, respectively. We have checked that this scaling holds as one performs calculations in larger nuclei or as one increases the model space. Average computational efficiency per processor increases from 10% for O-16 to 25% for 40-Ca at fixed processor number, indicating that the current algorithm better utilizes processor power and memory bandwidth during the tensor-multiplies for heavier nuclei. We believe an algorithmic jump will be required to get to 10K processors. This is currently under active investigation. Large runs today use up the maximum memory per processor available. The code scales to about 1000 processors at present. This is both a flops and memory problem. The secondary aspect, treating scattering problems within CC theory will also require some innovation although we are well on the way there. We recently transformed the CC code so that it can include complex basis states (required for the scattering problem). Order of magnitude scaling notes (N = number of basis states, n = number of particles): for NUCCOR code flop scaling is $O\{(n^2(N-n)^4\}$; for NUCCOR, code memory needs are $O(N^4)$ (interaction) + $O\{(n^2(N-n)^2\}$ (amplitudes). Current status: N = 480, n = 16–40. (O-16 and 40-Ca), and n = 4–8 (He chain). Desired status: N = 1000, n = 40–100 (NSLER deliverable); scale-up from present: memory (today: N = 480, n = 16): 425 Gbytes (interaction) and 2 Gbytes × 5 arrays; memory (3 years: N = 1000, n = 100): 8 Tbytes (interaction) and 64 Gbytes × 5 arrays;

Ops: today (N = 480 and n = 16): 9×10^{12} for 1 iteration (takes 20 for single solution); Ops: (3 years, N = 1000, n = 100): 6×10^{15} for 1 iteration (takes 20 for single solution).

If Chosen for Science Day One

The long-term goal of the project is to compute from first principles the properties of medium mass nuclei (mass 40–100) with two- and three-body nuclear forces included using the coupled-cluster method. This will enable an ab initio understanding of both nuclear properties and nuclear reaction mechanisms. We will achieve a quantitative understanding of nuclei and nuclear forces and will obtain theories (and hence simulations) of the nuclear quantum many-body problem that will enable systematic improvements to the desired degrees of accuracy and predictive capability for both nuclear properties and scattering cross sections relevant to several applications in SC/NP, NNSA, and the future GNEP programs. Some exciting quantum many-body science could come out. The ability to treat medium-mass nuclei from an ab initio point of view has never been a possibility until now and this means that the new systems *will* produce new and exciting science.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
F90	MPI	MPI-IO (essential)	None

Libraries and Tools

Library	Function	Functionality
BLAS		Matrix-matrix; matrix-vector
BLAS		tensor-tensor multiplies of size $100^2 \times 100^2$ and 1000^4 (100 particles and 1000 basis states)

Code Reference

David Dean (deandj@ornl.gov)

D. J. Dean and M. Hjorth-Jensen, "Coupled-Cluster Approach to Nuclear Physics," *Physical Review C* **69**, 054320 (2004).

NCCS Point of Contact

Edoardo Apra

aprae@ornl.gov

NWChem

Physics Models

NWChem provides many methods to compute the properties of molecular and periodic systems using standard quantum mechanical descriptions of the electronic wave function or density. In addition, NWChem has the capability to perform classical molecular dynamics and free energy simulations. These approaches may be combined to perform mixed quantum mechanics and molecular mechanics simulations.

Algorithms

NWChem uses both local basis function (atomic orbitals) and plane waves to compute the solution of the Schrödinger equations.

Scaling

Since NWChem is made of different modules, each module has its own different scalability features.

Various modules have showed scalability up to 300–500 processors (see density functional theory [DFT] scalability plot in Fig. E.11). A recent NUCCOR(T) calculations was run on 1400 processors EMSL-MSCF showing 60% of the aggregate peak floating-point performance.

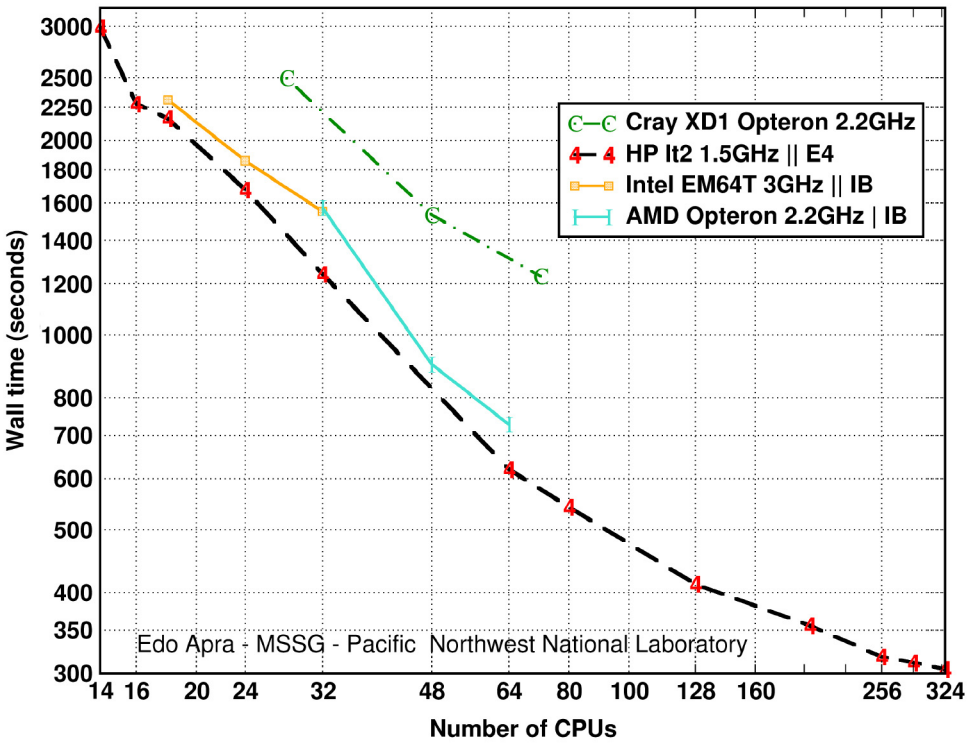


Fig. E.11. Benchmarks of the DFT code on various architectures.

If Chosen for Acceptance

NWChem uses the Global Arrays library for the bulk of its communications. The global arrays use the aggregate remote memory copy interface (ARMCI) library as a run-time system; therefore, an efficient port of ARMCI on the XT series is required to get good parallel scaling of NWChem.

NWChem relies on the ChemIO library both for parallel and serial I/O, and this requires an efficient ChemIO port on the XT series file system. Some software development effort is most likely needed to get the code to scale at O(10K) processors.

Runtime (in wall clock time) for various fixed-size problems at various node counts are acceptance metrics. The correctness of results must be checked against reference results (possibly from alternative hardware architectures).

If Chosen for Science Day One

Electronic structures calculations on large carbon nanotubes and metallic nanoparticles at various levels (DFT, mathematics-physics platform [MP2] and coupled cluster [CC]) will be performed. The larger the molecular aggregate, the more likely the simulation would be to represent data to be compared with experiment. These calculations will allow studying phenomena pertinent to catalysis and nanotechnology.

Since the DFT, MP2 and CC have a different kind of scaling with respect of system size—with DFT being the most affordable method, CC the most expensive, and MP2 of intermediate cost—simulations on the largest molecular aggregates will be feasible only with DFT; CC would be restricted to use for smaller molecular sizes. Quantities calculated will include energetics, structural, and vibrational properties.

As described here, a certain amount of software development effort is needed before the code performs at an efficient scale on $O(10^3)$ processors.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
GNU make, FORTRAN77/C	Global Arrays, ARMCI	ChemIO	

Math Libraries

Library	Function	Functionality
PEIGS		Symmetric eigensolvers, Cholesky decomposition
SCALAPACK		Symmetric eigensolvers, Cholesky decomposition, Linear solvers
LAPACK		Various dense linear algebra operations.
BLAS		Various dense linear algebra operations
FFTPACK		Discrete FFT

Code Reference

Edoardo Apra (aprae@ornl.gov)

http://www.emsl.pnl.gov/docs/nwchem/nwchem.html

NCCS Point of Contact

Edoardo Apra

aprae@ornl.gov

PFLOTRAN

Physics Models

PFLOTRAN (Parallel FLOw and TRANsport) solves multiphase, multicomponent reactive flow and transport equations in nonisothermal, variably saturated media. The code consists of two modules, which can be run separately or in coupled mode. The module PFLOW simulates Darcy flow, solving mass conservation equations for water and other fluids and an energy balance equation. The module PTRAN solves mass conservation equations for a multicomponent geochemical system. The reactions included in PTRAN involve aqueous species and minerals and can be written in the general form $\sum_j \nu_{ji} A_j \Leftrightarrow A_i$ and $\sum_j \nu_{jm} A_j \Leftrightarrow M_m$, respectively, where the set $\{A_j\}$ refers to a set of primary or basis species in terms of which all other species are written, A_i denotes an aqueous complex referred to as a secondary species, M_m refers to a mineral, and ν_{ji} and ν_{jm} are reaction stoichiometric coefficients derived from an extensive database.

Algorithms

PFLOTRAN uses a first-order finite-volume discretization on a Cartesian grid (extension to unstructured grids is being developed). Within both the PFLOW and PTRAN modules, time-stepping is fully implicit (backward Euler). In coupled mode, flow velocities, saturation, pressure, and temperature computed from PFLOW are fed into PTRAN. For transient problems, sequential coupling of PFLOW and PTRAN enables changes in porosity and permeability due to chemical reactions to alter the flow field.

A PETSc-based Newton-Krylov solver framework is used to solve the system of nonlinear equations arising at each time step. Because we employ PETSc, a wide variety of nonlinear and linear solver options can be easily employed by making the appropriate selection for the given problem at runtime. We usually employ an outer, quasi-Newton solver with line search and an inner, BiCGSTAB Krylov solver preconditioned with an additive-Schwarz method with an overlap of 1, with ILU(0) applied on each subdomain. The Jacobian matrix can be explicitly calculated (analytically for some cases, via finite-difference for others) or its action can be applied on the fly (though this somewhat restricts choice of preconditioners).

Adaptive mesh refinement (AMR) is currently not supported; we plan to use the Chombo framework to introduce support for hierarchical block-structured AMR.

Scaling

The current version of PFLOTRAN has exhibited linear (strong) scaling on up to 2048 processors on Jaguar and good (though nonlinear) scaling to 4096 processors (Fig. E.12). This is for a relatively modest one-phase thermo-hydrologic benchmark problem on a $25 \times 64 \times 256$ grid with three degrees of freedom per node (approximately 12.6 million degrees of freedom total). Simulations incorporating multiple phases or chemical reactions would probably exhibit excellent scaling to larger numbers of processors.

If Chosen for Acceptance

A functionality test would be quite useful, because PFLOTRAN exercises a large portion of the PETSc code base, which is employed by several codes. Code needs to be able to run at scale on a suite of benchmark problems and produce correct results.

If Chosen for Science Day One

PFLOTRAN has already been used to study uranium transport problems at the Hanford 300 site, radionuclide migration at the Nevada Test Site, and subsurface CO_2 sequestration. Problems from any of these sites could benefit from a great increase in CPU power, allowing better resolution of transient flow features, chemical reactions, and more detailed chemistry (with more chemical species). Further consultation with subsurface scientists is required to determine specific problems to propose. Working

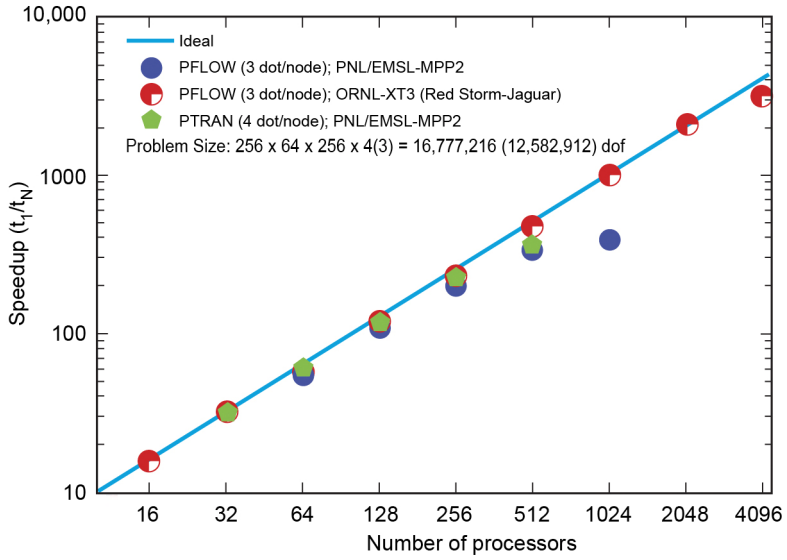


Fig. E.12. PFLOTRAN has exhibited linear (strong) scaling on up to 2048 processors on Jaguar and good (though nonlinear) scaling to 4096 processors. (Richard Mills, ORNL and Peter Lichtner, LANL)

with LANL DOE Regional Partnerships on CO_2 sequestration, we will identify a field site and apply PFLOTRAN to perform multiscale, multiphase, multicomponent modeling of a 3-D field CO_2 injection scenario. We will include the presence of an oil phase and four-phase liquid-gas-aqueous-oil system to describe dissipation of the supercritical CO_2 phase and escape of CO_2 to the surface. We are particularly interested in resolving viscous fingering effects that result from buoyancy effects caused by an increase in density as supercritical CO_2 dissolves into the formation brine. Finger widths may be on the order of meters or smaller depending on the reservoir properties. Better understanding of these fingering phenomena can result in more effective and economical sequestration as well as enhanced oil recovery.

PFLOTRAN is *already* usable for solving real science problems on thousands of processors. Development is actively ongoing, and the types of problems that could be attempted for "Science Day One" depend very much on when "Day One" happens.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Fortran 90	MPI	None	None

Math Libraries

Library	Function	Functionality
PETSc	SNESSolve, KSPSolve, DAGlobalToLocal, MatFDColoring	Newton solves, Krylov solves, halo exchanges, multi-color finite difference Jacobian
BLAS	BLAS Level 1 and 2	Dot product, etc.

Code Reference

Peter Lichtner (lichtner@lanl.gov)

R. T. Mills et al., "Simulating Subsurface Flow and Transport on Ultrascale Computers Using PFLOTRAN," *Journal of Physics Conference Series* **78**, 012051 (2007).

NCCS Point of Contact

Richard Mills

rmills@ornl.gov

POP/CICE

Physics Models

POP is an ocean circulation model derived from earlier models in which depth is used as the vertical coordinate. The model solves the three-dimensional primitive equations for fluid motions on the sphere under hydrostatic and Boussinesq approximations. A wide variety of physical parameterizations and other features are available in the model and are described in detail in a reference manual distributed with the code. Because POP is a public code, many improvements to its physical parameterizations have resulted from external collaborations with other ocean-modeling groups, and such development is very much a community effort.

The Los Alamos Sea Ice Model (CICE) features the energy conserving thermodynamics model with four layers of ice and one layer of snow in each of five ice-thickness categories, the energy-based ridging scheme, an ice strength parameterization, elastic-viscous-plastic ice dynamics, and horizontal advection via a new incremental remapping scheme. Prognostic variables for each thickness category include ice area fraction, ice volume, ice energy in each vertical layer, snow energy, and surface temperature. A nonlinear, vertical salinity profile remains constant. The temperature dependence of the longwave radiation and sensible and latent heat fluxes is included in the nonlinear flux balance that (iteratively) determines the ice or snow surface temperature. The albedo parameterization depends on surface type (snow or bare ice), surface temperature (not just whether it is melting or frozen), and both ice and snow thickness. Ice and snow albedo values are merged based on a snow "patchiness" fraction. The ice model can accommodate four wavelengths of radiation and thus have four associated albedos; with just one

wavelength available for forcing, the four albedos are weighted and merged into a single value. For more details, including a full set of references, see the model documentation.

Algorithms

Spatial derivatives in both POP and CICE are computed using finite-difference discretizations which are formulated to handle any generalized orthogonal grid on a sphere, including dipole and tripole grids which shift the North Pole singularity into land masses to avoid time-step constraints due to grid convergence.

Time integration of the POP model is split into two parts. The 3-D vertically varying (baroclinic) tendencies are integrated explicitly using a leapfrog scheme. The very fast vertically-uniform (barotropic) modes are integrated using an implicit free surface formulation in which a preconditioned conjugate gradient solver is used to solve for the 2-D surface pressure. CICE is integrated in time using fully explicit methods.

Scaling

Although POP was originally developed for the Connection Machine, it was designed from the start for portability by isolating all routines involving communication into a small set (5) of modules which can be modified for specific architectures. Currently, versions of these routines exist for MPI and SHMEM communication libraries and also for serial execution. The appropriate directory is chosen at the time it is compiled, and no pre-processor directives are used to support different machines. Support for hybrid programming using threads and message passing has recently been added and is described in the user's guide.

POP tends to be compute-bound as long as the number of cells per processor is high enough to swamp a latency-bound, 2-D elliptic solve.

The 0.1° problem scales to the full size of single-core Jaguar, but latency in the barotropic will soon dominate. Early tests on the dual-core Jaguar show that POP is having a problem with MPI wait times (see Figs. E.13 and E.14). Tuning of MPI and virtual-node Catamount will probably be necessary.

If Chosen for Acceptance

A standard benchmark configuration of stand-alone POP is available and could be used for a performance-based acceptance test. The problem size should be 0.1°. We have some historic data for the displaced-pole grid, but current production runs use the tripole grid. Acceptance metrics are (1) runtime (in simulated years per CPU day) for a fixed-size problem at various node counts and (2) lower bounds on per-socket performance.

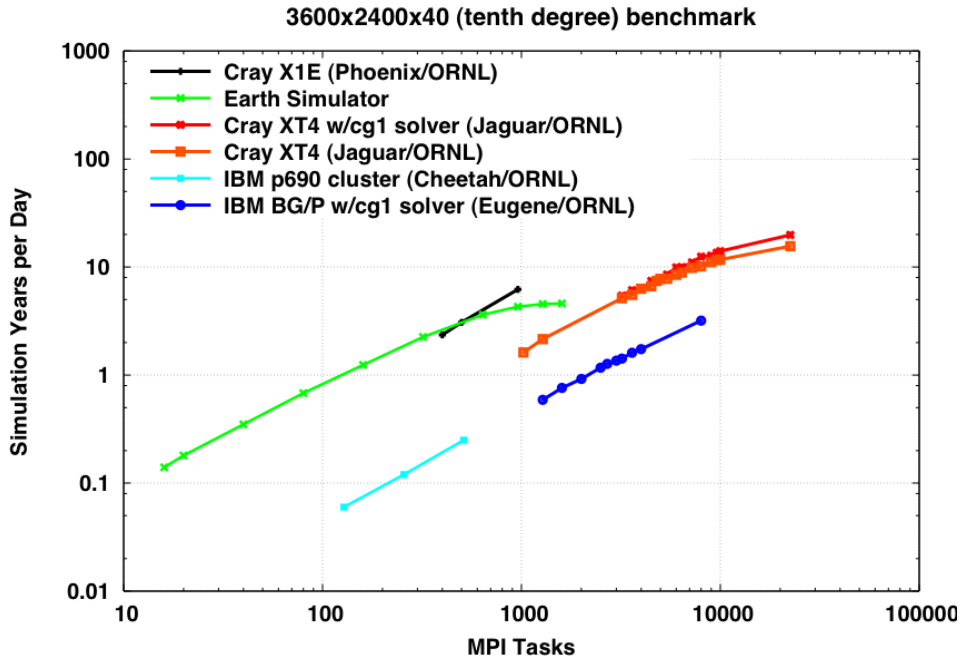


Fig. E.13. Parallel Ocean Program (POP) 1.4.3: 0.1-degree benchmark, logarithmic axes.

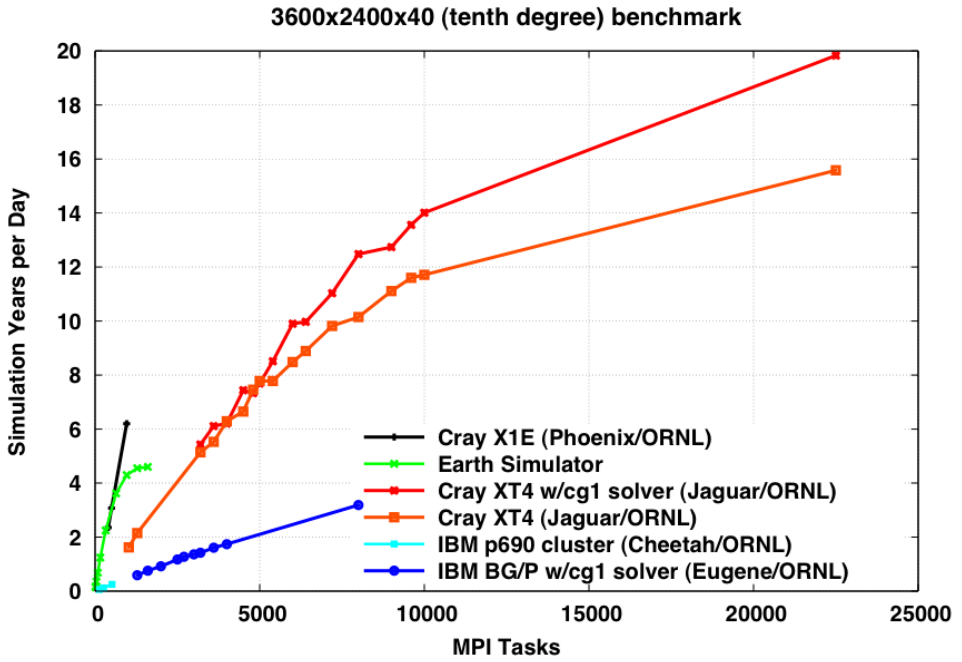


Fig. E.14. Parallel Ocean Program (POP) 1.4.3: 0.1-degree benchmark, linear axes.

If Chosen for Science Day One

Eddy-resolving ocean simulations have been shown to be necessary for the accurate representation of the ocean circulation. To date, such simulations have been done with the ocean model only with a focus on the surface wind-driven circulation and at time scales of decades. Current coupled model simulations use coarser resolution ocean configurations because of the computational expense of an eddy-resolving ocean in century-scale ensemble simulations. A fully-coupled experiment with an eddy-resolving ocean and sea ice is necessary to reduce some of the coupled model biases, particularly in the North Atlantic, where the ice extent and deep water formation are governed by the accurate representation of the North Atlantic current systems.

After a short spinup of the ocean-ice system at 0.1° resolution, a fully-coupled CCSM configuration with T85 CAM and 0.1 degree POP and CICE will be integrated for a few decades under a rapid CO_2 doubling scenario to evaluate the model in both current and future climate change regimes.

For a fixed-size problem, POP scaling is limited by the latency-dominated conjugate gradient (CG) solves in the barotropic computation. Improvements to the solver could reduce the number of iterations and allow higher scalability. Use of OpenMP to reduce the number of MPI tasks could also help scalability, but tuning of the OpenMP in POP may be necessary.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Fortran 90, C, GNU Make, CAF (optional)	MPI, OpenMP (optional)	NetCDF	None

Math Libraries

Library	Function	Functionality
None		Inline sparse linear solve (CG) could be a library call

Code Reference

Phil Jones (pwjones@lanl.gov)

http://climate.lanl.gov/Models/POP/

NCCS Point of Contact

James B. White III

trey@ornl.gov

Qbox

Physics Models

This model describes first-principles molecular dynamics within density functional theory.

Algorithms

The algorithm used is the plane-wave, pseudopotential method.

Scaling

The scaling efficiency of Qbox was demonstrated on various parallel systems including MCR, Thunder and BlueGene/L at LLNL, using up to 131,072 CPUs. Examples of scaling are summarized in Tables E.4 and E.5 and Fig. E.15.

Table E.4 shows the performance of the Qbox code for a calculation of the electronic structure of a Cd33Se33 nanoparticle. The parallel efficiency is 97% between 128 and 210 CPUs and 82% between 128 and 420 CPUs. Results were obtained on MCR, a 1152-node dual Xeon/Quadrics cluster installed at the LLNL.

Table E.4. Performance of the Qbox code for a calculation of the electronic structure of a Cd33Se33 nanoparticle

Nodes	CPUs	Time/step	Speedup	Efficiency
64	128	43	1.00	1.00
105	210	27	1.59	0.97
210	420	16	2.69	0.82

Table E.5. Performance of the Qbox code for a calculation of the electronic structure of 512 H_2O molecules

Nodes	CPUs	Time/step	Speedup	Efficiency
280	1120	46	1.00	1.00
560	2240	25	1.84	0.92
908	3920	16	3.07	0.88

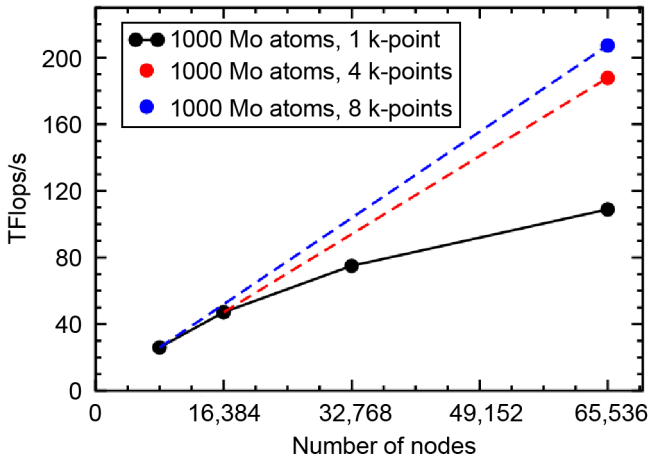


Fig. E.15. Strong scaling Qbox results on BlueGene/L for 1000 molybdenum atoms with 1 (non-zero) k-point. Also shown is the sustained performance on the full machine (64K nodes) with multiple k-points. Dashed lines indicate perfect scaling between the measured full machine result and the equivalent individual k-point calculations.

Table E.5 shows the performance of the Qbox code for a calculation of the electronic structure of 512 H_2O molecules. The parallel efficiency is 92% between 1120 and 2240 CPUs and 88% between 1120 and 3920 CPUs. Results were obtained on Thunder, a 1024-node quad Itanium2/Quadrics cluster installed at LLNL.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
C++	MPI		

Math Libraries

Library	Function	Functionality
BLAS, LAPACK	All	Linear algebra
BLACS, ScaLAPACK	All	Parallel linear algebra
FFTW	All	1-D Fourier transforms
Apache Xerces-C	All	XML parsing

Code Reference

Francois Gygi (fgygi@ucdavis.edu)

http://eslab.ucdavis.edu/software/qbox/index.htm

NCCS Point of Contact

Doug Kothe

kothe@ornl.gov

QMC/DCA

Physics Models

The two-dimensional Hubbard model is a simplified description of the electronic degrees of freedom of the superconducting copper-oxide planes in high-temperature superconductors (HYSC). Despite its simplicity, it is believed to hold the key ingredients necessary to explain the phenomenon of high-temperature superconductivity. The QMC/DCA code is based on a dynamic cluster quantum Monte Carlo algorithm to solve lattice models of strongly correlated electron systems such as the 2-D Hubbard model in a controlled way. The dynamic cluster method approximates the effects of correlations in the bulk lattice with those of a finite-size quantum cluster. This enables a mapping of the bulk lattice problem to an effective cluster embedded in a self-consistent bath designed to represent the remaining degrees of freedom. Recently, this technique has been applied successfully to show that the 2-D Hubbard model of high-temperature superconductors does have a superconducting transition in the range of parameters and temperatures characteristic of the cuprates. The new computational capabilities even established the fact that pairing in the Hubbard model is mediated by spin fluctuations. While the success in describing the physics of the cuprates with high-end simulation results of the Hubbard model is remarkable, it is important to link a generalized Hubbard-like model to actual cuprate HTSC to understand material-specific properties such as the huge differences in superconducting transition temperatures between different HTSC materials. This project will require the solution of a multiband Hubbard model with possibly more than one correlated band.

Algorithm

The computational workhorse to solve the effective quantum cluster problem is a generalized version of the Hirsch-Fye QMC algorithm. This algorithm performs a stochastic Markov-chain walk, along which measurements are made periodically. The central quantity that has to be measured and updated along this walk is the single-particle Green's function G of the effective cluster problem. G is a matrix of size $N*t$, where N is the total number of sites and orbitals treated with correlations in the quantum cluster calculation and t is the number of time-slices used in the integration path integral. A majority of the CPU

time is spent updating G that is calculated by a vector outer product followed by a matrix update, which may be completed by the BLAS call DGER. Since DGER has a relatively low computational intensity (only two floating point operations per memory access), a reformulation of the underlying Hirsch-Fye algorithm is used, in which the frequent calls to DGER are delayed and hence replaced by fewer and much more cache-efficient matrix multiplies (BLAS call DGEMM). This allows the QMC/DCA code to be run for large problems with high efficiency on superscalar processors. The measurements of additional four-point correlation functions are represented as complex matrix-matrix products and completed with the BLAS call CGEMM.

Scaling

The QMC algorithm is inherently parallel because the measurements made along the Markov-chain walk need to be independent. The code therefore performs several independent, shorter Markov-chain walks on different processors and averages the results of the individual walks to obtain the final result using MPI. Apart from the fraction of the walk required to achieve equilibrium, the result is an almost perfect parallel speedup with increasing number of processors because no communication between processors is required during the Markov process. As a result, the code scales perfectly to ~1000 processors or multicore sockets. Further scaling by a factor of 10 or more should be achievable by distributing one Markov chain over multiple sockets. Such hybrid parallelism will be called for because the size $(Nt)^2$ of the matrices in the matrix multiplies for the multiband problems will be large, and therefore the runtime may be reduced by distributing the matrix multiplies over many sockets.

If Chosen for Acceptance

The QMC/DCA code runs on a wide variety of architectures. For the acceptance test, it is possible to generate test runs of virtually any size with a known or cross-checkable answer that stress MPI, BLAS, LAPACK, and the F90 compiler.

If Chosen for Science Day One

We will perform material-specific simulations of high-temperature superconductors, using QMC/DCA simulations of multiband Hubbard models with realistic parameters determined with bandstructure methods.

Compared to the simulation of the single-band model that used, 25% of the LCF Cray X1E, a simulation of a three-band model will be a factor of 27 more expensive. This factor can possibly be as much as 200 if the number of time slices t has to be doubled because of an increase in the strength of the onsite Coulomb interaction parameters. Therefore, since the simulation of the single-band model is a

teraflop problem, the simulations of a materials-specific multiband model are a petascale computing problem.

To scale the QMC/DCA code efficiently to more than 1000 sockets, the code will be further parallelized by distributing the individual Markov-chain walks across multiple sockets. The effort associated with this modification is moderate.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Fortran 90	MPI	None	None

Code Reference

Thomas Schulthess (schulthesstc@ornl.gov)

A. Maier et al., "Quantum Cluster Theories," *Review of Modern Physics* **77**, 1027 (2005).

NCCS Point of Contact

Markus Eisenbach

eisenbachm@ornl.gov

S3D

Physics Models

S3D solves a fully coupled system of time-varying partial differential equations (PDEs) governing the full compressible reacting Navier-Stokes, total energy, species continuity and continuity equations coupled with detailed chemistry. The PDEs are supplemented with additional constitutive relationships for the ideal gas equation of state, and detailed high-fidelity models for reaction rate, molecular transport, and thermodynamic properties. A summary of this formulation follows:

After the initialization of the primitive variables for each time step the convective, diffusive and chemical terms in the conservation equations are updated, once for each of the six stages of the fourth-order accurate explicit Runge-Kutta time advancement solver. The main kernels in this solver where over 95% of the computation occurs are given below:

- **Chemistry**: Computes chemical reaction rate source terms for species equations. The chemical kinetics data are preprocessed and the code to compute the reaction rates, named as "getrates," is generated by the Chemkin compatible preprocessing utility Autogetrates package. The routines are packaged in a separate module that acts as an interface to the code and abstracts the actual

implementation of the reaction rates computation. This will allow the use of different versions of the getrates subroutine targeted at different systems.

- **Transport:** Computes molecular transport properties for the species. The properties computed include the viscosity, thermal diffusivity, and species mass diffusivities. The code is linked with the transport library that is part of the standard Chemkin suite.
- **Thermodynamics**: Computes the thermodynamic properties, such as enthalpy and specific heats of the mixture. The thermodynamic data are given in the Chemkin compatible format and are preprocessed through the Chemkin interpreter (http://reactiondesign.com). Rather than directly evaluate the properties using the Chemkin routines, the code employs a tabulation and lookup strategy.
- **Derivatives**: Computes the spatial derivatives of the primitive and conserved variables using higher-order finite difference operators. The code uses nonblocking sends and receives to exchange the data at the processor boundaries among different processors.
- **Other RHS**: The right-hand side of the time advance equation involves all of the above-mentioned operations and the convection terms. These terms are summed up according to the governing equations. All operations involved in this procedure are lumped into the other RHS module for accounting purposes.
- **Time Integration**: Advances the solution in time using a fourth-order accurate Runge-Kutta scheme. This module also includes an error controller that routinely checks for the time accuracy of the solution and adjusts the time step to achieve the desired error tolerances.

ALGORITHMS

S3D is based on a high-order accurate, non-dissipative numerical scheme. It has been used extensively to investigate first-of-a-kind fundamental turbulence-chemistry interactions in combustion topics, including premixed nonpremixed flames and autoignition. Time advancement is achieved through a fourth-order explicit Runge-Kutta method, spatial differencing is achieved through high-order (eighth-order with tenth-order filters) finite differences on a Cartesian structured grid, and Navier-Stokes Characteristic Boundary Conditions (NSCBC) are used to prescribe the boundary conditions. The equations are solved on a conventional structured mesh.

This computational approach is very appropriate for the problems selected. The coupling of high-order finite difference methods with explicit R-K time integration make very effective use of the available resources, obtaining spectral-like spatial resolution without excessive communication overheads and allowing scalable parallelism.

SCALING

The parallelism in S3D can be basically described as explicit nearest-neighbor local communication (Fig. E.16). With this design, the code is compute-bound, which has been empirically observed. The scaling of the code is demonstrated with a weak-scaling test; that is, as the processor count increases so does the total amount of work (i.e., the work per process stays constant.) The scaling efficiency on the Cray XT series has been observed to be over 90% for up to 5120 processors.

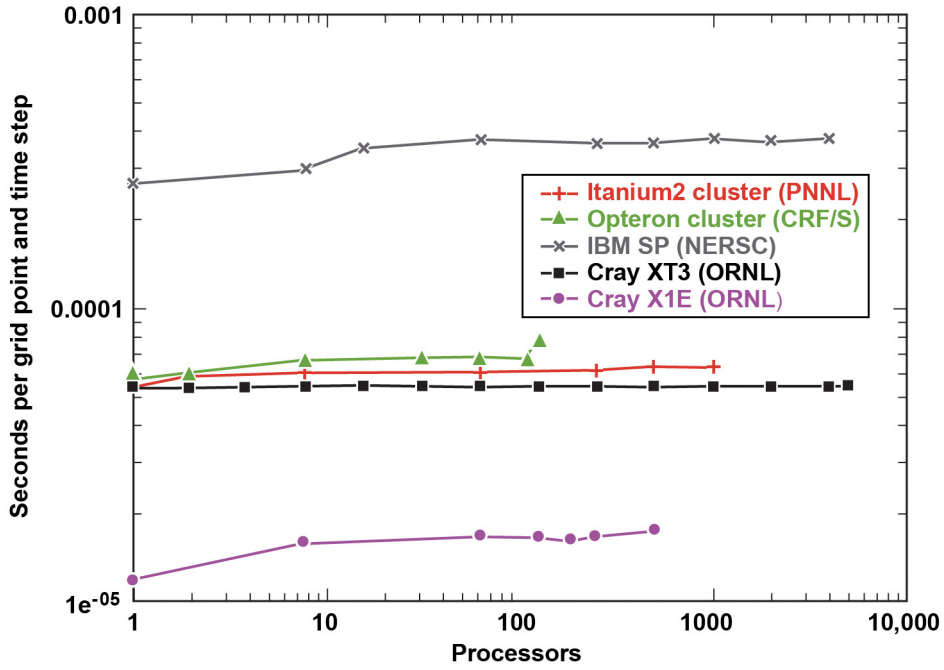


Fig. E.16. S3D scaling demonstrated with a weak-scaling test.

IF CHOSEN FOR ACCEPTANCE

Acceptance test criteria for S3D are

- Simulation with realistic I/O works/produces correct answers,
- Capable of running correctly at a variety of processor counts from 1 up to maximum size of machine, and
- (For upgrades only) the flame benchmark time/step/gridpoint rate remains the same (within 5%) or gets shorter.

IF CHOSEN FOR SCIENCE DAY ONE

Flame stabilization in low temperature mixing-controlled diesel combustion: a Petascale simulation. Diesel combustion has the potential to be a "game changer" for energy surety because diesels are 30–40% more efficient than comparable gasoline engines. But to gain acceptance, diesels must be cleaner. Successfully meeting upcoming ultra-low emission standards could mean widespread adoption of diesels and major reductions in foreign oil dependency. Low temperature mixing-controlled diesel

combustion has shown promise in meeting NO_x and soot emission standards and is easier to control than HCCI. However, optimization requires accurate understanding and modeling. Alongside experiments in an optically accessible combustion vessel at Sandia National Laboratories' Combustion Research Facility, we propose to perform high fidelity DNS of high-pressure, n-heptane jets. With access to the full unsteady thermo-chemical fields from the DNS, augmented by chemiluminescence diagnostics, we will study mechanisms for lift-off stabilization and formation of key soot precursors. An important outstanding question is whether or not the lift-off stabilization is supported by premixed flame propagation into autoigniting cool flame mixture or by transition to second-stage chemistry through self-ignition. Cool flame autoignition is expected to have a strong effect on flame speed and may support flame propagation even at very low flame temperature. Alternatively, cool flame (low-temperature ignition) activity may also lead directly to second-stage reaction in the absence of flame propagation. More detail is needed to understand the importance of flame propagation versus autoignition in the high-temperature reaction zone at the lift-off length. This information can uniquely be obtained from DNS with detailed n-heptane chemical kinetics.

This set of simulations would require the use of a stiff explicit-implicit time integrator which we have developed known as the additive fourth-order Runge-Kutta method. Alternatively, the chemical kinetic mechanism for n-heptane spanning low, intermediate and high-temperature kinetics would need to have the stiffness removed. Our preliminary estimates suggest that the grid resolution required to resolve the ignition fronts at high pressure is small (in the order of microns). Therefore, compared to our atmospheric flames, the number of grids will be expensive. Also, the runs need to be simulated to ignition delay times of several milliseconds. This run will require petascale computing to be feasible in three-dimensions.

Stabilization mechanisms in lifted, vitiated flames: a 250-TF simulation. In many modern combustion systems, fuel is injected into an environment of hot gases, and a flame may be stabilized through the recirculation of hot air and combustion products. Under many conditions, this leads to a lifted flame, and the hot environment admits the possibility of autoignition as a mechanism of stabilization of the flame base. Clear understanding of turbulent flame stabilization in an environment of hot combustion products will aid the advancement of combustion technology, including modern recirculation burners, and understanding of possible auto-igniting stabilization modes may be also relevant to diesel engines.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Fortran 90	MPI	None	Tar, Cp, Mkdir, Rm (optional)

Code Reference

Jacquelin Chen (jhchen@sandia.gov)

E. R. Hawkes et al., "Direct Numerical Simulation of Turbulent Combustion: Fundamental Insights Towards Predictive Models," *Journal of Physics: Conference Series* **16**, 65–79 (2005).

NCCS Point of Contact

Ramanan Sankaran

sankaranr@ornl.gov

T3P

Physics Models

T3P is one of a suite of codes used in the design of a low-loss accelerating cavity for the International Linear Accelerator (ILC). T3P solves Maxwell's equations, defined on a 3-D unstructured grid, via a finite element discretization using basis functions of up to the sixth order. This is used to describe the transit of a particle beam through an accelerating cavity and under the proper boundary conditions to calculate the longitudinal and transverse wakefields.

Algorithms

T3P is based on a finite element method using basis functions of up to the sixth order, using the implicit Newark-Beta method for time stepping. The resulting sparse matrices are real, symmetric positive definite. With a good mesh, convergence typically requires about 200 iterations using CG with incomplete Cholesky preconditioning. However, this can significantly increase if the mesh get badly distorted, which results in poorly conditioned systems.

Sparsity is a function of the order of the basis functions: first-order results in about 15 nonzero elements per row; second-order, about 50 nonzero elements per row; third-order are a little denser. In terms of the pattern of the nonzero elements, using second-order basis functions will have two-by-two dense blocks; using third-order basis functions will have 3×3 blocks and 6×6 blocks. The choice of method, direct or iterative, involves a trade-off between speed (direct solver) and memory requirements (sparse solver).

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
C++	MPI	netcdf	–

Math Libraries

Library	Function	Functionality
Pnetcdf	nc_<open,put_xy,inq_xy,close>	I/O
ParMetis	PARKMETIS, PARRMETIS, PARUAMETIS, PARKMETIS,	Discretized domain partitioning (preferred)
Zoltan	Zoltan_Create, Zoltan_Set_Param, Zoltan_Set_Fn, Zoltan_Destroy, Zoltan_LB_Balance, Zoltan_LB_Eval	Discretized domain partitioning
MUMPS	dmumps_c,cmumps_c, zmumps_c,smumps_c	Direct solver of sparse linear systems.
ScaLAPACK	MUMPS dependency	Linear system solver; called by MUMPS.
SLAC code proper	Conjugate gradient method, with various preconditioners: incomplete Cholsky, hierarchical methods, etc.	Sparse linear system solver option.

Code Reference

Rich Lee (liequn@slac.stanford.edu)

K. Ko et al., "Advances in Electromagnetic Modeling Through High Performance Computing," *Physica C* **441** 258–262 (2006).

NCCS Point of Contact

Richard Barrett

rbarrett@ornl.gov

VASP (+WL)

Physics Models

Plane wave-based density functional calculations, together with all-electron-derived pseudo potentials, are a powerful and flexible method. Their well-controlled accuracy vs. computational cost makes them ideal for the study of novel systems in which the electronic structure is not well understood,

or in which tiny differences determine the outcome of the simulations. Such accuracy is critical when performing quantum molecular dynamics (QMD) simulations, which enable studies of the evolution of nanoscale systems and their environment at finite temperature, as well as investigations of biomolecular reaction mechanisms, structural changes and temperature-dependent phase transitions. Although the method is in-principle cubic scaling, in practice it scales quadratically up to 1000 atoms using recent numerical advances.

Algorithms

Planewave codes density functional codes solve the density functional equations in a plane wave basis defined by a sphere of vectors in Fourier space. All atoms are represented by ab initio pseudopotentials, of either a norm-conserving, ultrasoft, or projector-augmented wave type. The latter two offer much improved accuracy and reduced computational costs (flops and memory) over the simpler norm-conserving potentials, particularly for systems containing transition metal atoms. VASP implements all of these options at the expense of complexity, whereas, for example, the Qbox code only implements norm-conserving potentials. For calculations of up to 1000 atoms, the main computational effort involves (1) evaluation of the pseudopotential contributions to the energy and forces, and (2) parallel Fourier transforms between real and reciprocal (Fourier) space. The former involve linear algebra operations using standard BLAS, while the latter utilize vendor 1-D FFT transforms and custom routines for highly efficient parallel 3-D transforms. Appropriately configured, VASP currently delivers a large fraction of peak performance, typically 30–50%, up to 1000 processors.

Scaling

VASP presently scales to about 1000 processors for a system of hundreds of atoms. An optimized version on the Cray X1/X1E, achieves over 1 TF for a prototypical 807 atom FePt atom nanoparticle on 128–512 multi-streaming vector processors MSPs. Experience with other codes suggests that hard scaling can be improved by at least one order of magnitude if the parallel FFTs and some global MPI operations are carefully optimized. Soft scaling should allow systems of several thousand atoms to be studied.

If Chosen for Acceptance

VASP is a very robust code that runs on a wide variety of architectures. It has not exotic library dependencies and could readily form part of an acceptance or performance test. It is possible to generate test runs of virtually any size with a known or cross-checkable answer that stress MPI, BLAS, SCALAPACK, and F90 compiler.

If Chosen for Science Day One

A combined VASP+gWL code will allow the study of the chemical phase diagram of FePt nanoparticles as a function of size. Not much is presently known about the chemical order in these nanoparticles, information that seems crucial for the understanding of magnetic properties. Knowing the temperature- and size-dependent free energy and phase diagram of these particles could give important experimental guidance for synthesis. Such detailed sub-nanoscale information is almost impossible to extract from experiment alone and computational results will play a key role. The achievability of these results will depend on the number of WL samples needed to compute free energies to sufficient accuracy.

Functional Software Requirements

System Software

Programming languages	Communication libraries	I/O libraries and functions	Operating system functions
Fortran90 C (minor utility only)	MPI	None	Getrusage (easily changed)

Math Libraries

Library	Function	Functionality
BLAS	ZGEMM, DGEMM	Double complex/real general matrix-matrix multiply
BLAS	ZTRMM, DTRMM	Double complex/real triangular matrix-matrix multiply
SCALAPACK	PDTRTRI PZTRTRI	Matrix inverse, Cholesky decomposition, Eigenvector computation
	PDPOTRF PZPOTRF	
	PZHEEVX PDSYEVX	

Code Reference

Paul Kent (kentpr@ornl.gov)

http://cms.mpi.univie.ac.at/vasp/

NCCS Point of Contact

Markus Eisenbach

eisenbachm@ornl.gov


OAK
RIDGE
National Laboratory