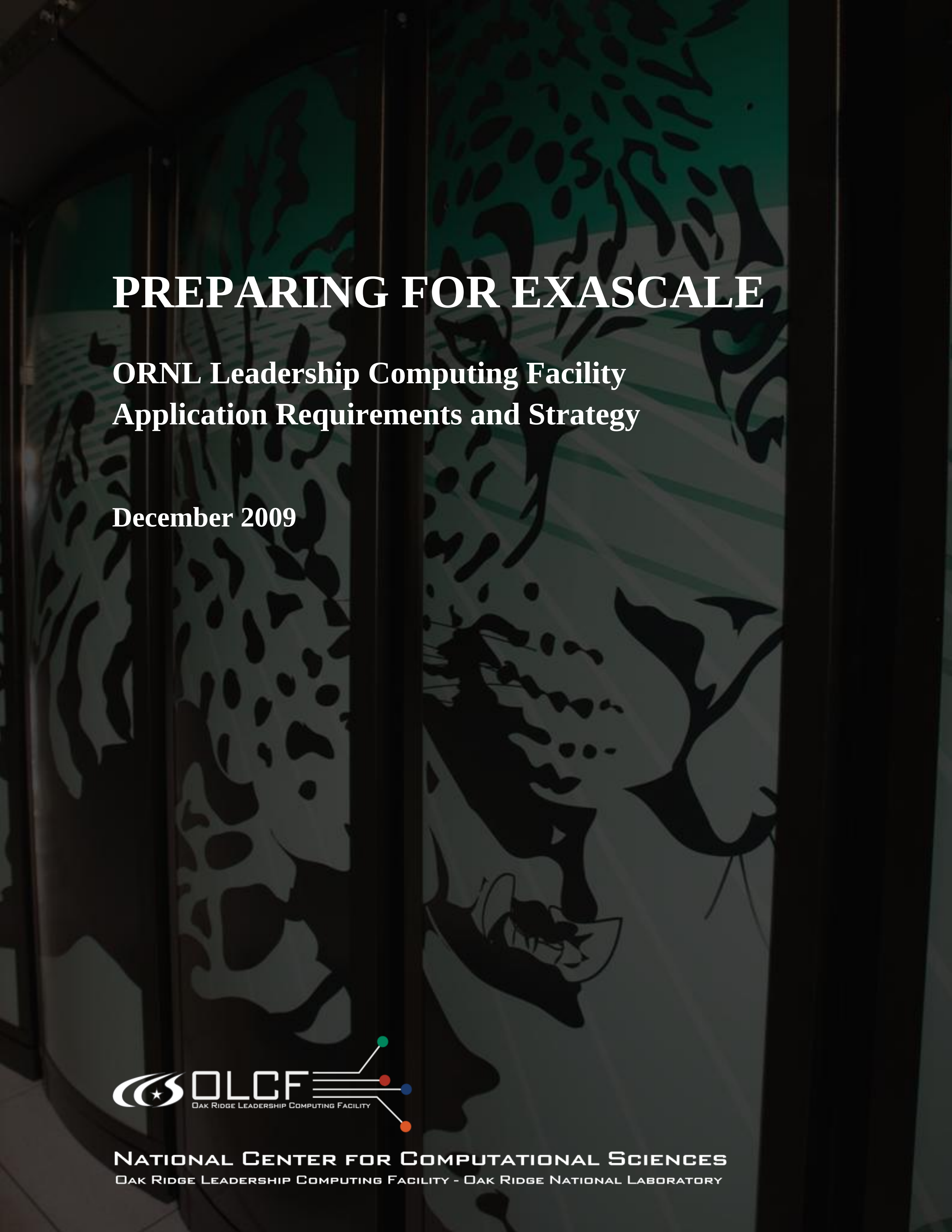


# PREPARING FOR EXASCALE

**ORNL Leadership Computing Facility**
**Application Requirements and Strategy**

**December 2009**

OLCF
Oak Ridge Leadership Computing Facility

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

ORNL/TM-2009/308

Oak Ridge Leadership Computing Facility
National Center for Computational Sciences

# PREPARING FOR EXASCALE: ORNL Leadership Computing Facility Application Requirements and Strategy

Wayne Joubert
Douglas Kothe
Hai Ah Nam

**Date Published: December 2009**

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

# TABLES

# FIGURES

# EXECUTIVE SUMMARY

In 2009 the Oak Ridge Leadership Computing Facility (OLCF), a U.S. Department of Energy (DOE) facility 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. This effort targeted science teams whose projects received large computer allocation awards on OLCF systems.

A clear finding of this process was that in order to reach their science goals over the next several years, multiple projects will require computational resources in excess of an order of magnitude more powerful than those currently available. Additionally, for the longer term, next-generation science will require computing platforms of exascale capability in order to reach DOE science objectives over the next decade.

It is generally recognized that achieving exascale in the proposed time frame will require disruptive changes in computer hardware and software. Processor hardware will become necessarily heterogeneous and will include accelerator technologies. Software must undergo the concomitant changes needed to extract the available performance from this heterogeneous hardware. This disruption portends to be substantial, not unlike the change to the message passing paradigm in the computational science community over 20 years ago.

Since technological disruptions take time to assimilate, we must aggressively embark on this course of change now, to insure that science applications and their underlying programming models are mature and ready when exascale computing arrives. This includes initiation of application readiness efforts to adapt existing codes to heterogeneous architectures, support of relevant software tools, and procurement of next-generation hardware testbeds for porting and testing codes.

The 2009 OLCF requirements process identified numerous actions necessary to meet this challenge:

- Hardware capabilities must be advanced on multiple fronts, including peak flops, node memory capacity, interconnect latency, interconnect bandwidth, and memory bandwidth.
- Effective parallel programming interfaces must be developed to exploit the power of emerging hardware.
- Science application teams must now begin to adapt and reformulate application codes to the new hardware and software, typified by hierarchical and disparate layers of compute, memory and concurrency.

- Algorithm research must be realigned to exploit this hierarchy.
- When possible, mathematical libraries must be used to encapsulate the required operations in an efficient and useful way.
- Software tools must be developed to make the new hardware more usable.
- Science application software must be improved to cope with the increasing complexity of computing systems.
- Data management efforts must be readied for the larger quantities of data generated by larger, more accurate science models.

Requirements elicitation, analysis, validation, and management comprise a difficult and inexact process, particularly in periods of technological change. Nonetheless, the OLCF requirements modeling process is becoming increasingly quantitative and actionable, as the process becomes more developed and mature, and the process this year has identified clear and concrete steps to be taken.

This report discloses (1) the fundamental science case driving the need for the next generation of computer hardware, (2) application usage trends that illustrate the science need, (3) application performance characteristics that drive the need for increased hardware capabilities, (4) resource and process requirements that make the development and deployment of science applications on next-generation hardware successful, and (5) summary recommendations for the required next steps within the computer and computational science communities.

# 1. INTRODUCTION

## 1.1 Context

The Advanced Scientific Computing Research (ASCR) program of the U.S. Department of Energy (DOE) has identified key computational science goals vital to addressing fundamental national and global concerns in domains such as energy assurance, environment and national security. This report details the conclusions of the Oak Ridge Leadership Computing Facility (OLCF) requirements elicitation process employed to identify specific resource requirements for attaining these goals.

Reaching next-generation science objectives requires computational resources several orders of magnitude beyond those currently available. A commonly referenced milestone is an exaflop system (capable of $10^{18}$ floating point operations per second), which by current trends will likely be built within the 2015-2020 timeframe. To reach exascale computing and beyond, major disruptive changes will be required in parallel computing hardware and software. Some of these changes are already apparent, but adapting to these developments will take conscious planning and purposeful action. This report identifies the supporting elements required for transformational science progress in the 3-5 year timeframe while concurrently preparing for leadership computational science at the exascale level.

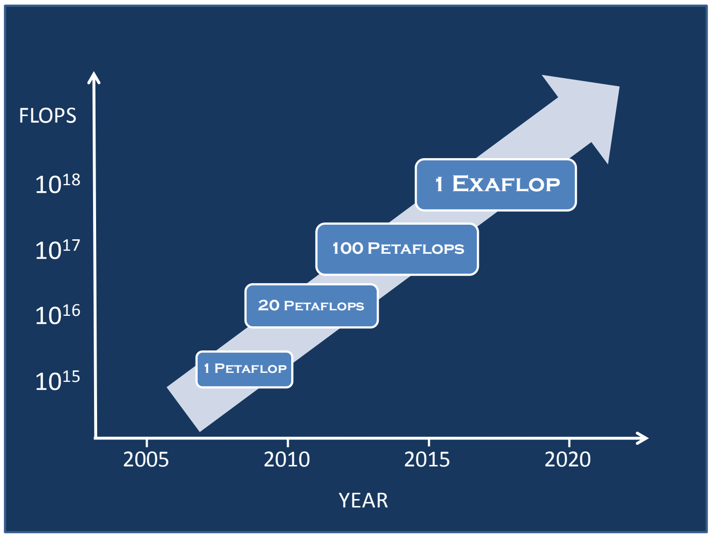


**Fig. 1.1 Predicted performance growth of leadership computing platforms**

## 1.2 Hardware Drivers

Historically, in the high performance computing (HPC) arena, technology trends and business factors have made the economics of HPC hardware heavily dependent on broader trends in the commodity hardware market. Planning for next-generation computational science must take into account the external environmental factor of availability of the required computing resources.

In recent years limitations in clock speed, power consumption and instruction level parallelism have led to the prevalence of multicore processors in the marketplace, in which a conventional processor core is replicated on a single die.  More recent developments however are trending toward processors that are many-core and heterogeneous, in which one or more conventional cores are augmented with multiple streaming processor cores.  This new direction in processor design is likely to radically impact parallel scientific computing. "Recent activities of major chip manufacturers, such as Intel, AMD, IBM and NVIDIA, make it more evident than ever that future designs of microprocessors and large HPC systems will be heterogeneous in nature" [Agullo et al. 2009].

The typical processor of the future will likely be composed of multiple processor cores of a more conventional nature supplemented by an attached set of multiple SIMD-like stream processing units used to offload local compute-intensive work.  Programming these processors to exploit potential performance gains will require significant new software approaches.  These changes will likely substantially affect parallel scientific applications.  It has been noted that "many familiar and widely used algorithms and libraries will become obsolete and will have to be rethought and rewritten in order to take advantage of the new architectures" [Tomov et al. 2008, p. 1].  As a result, "many of the tools, software, algorithms, and libraries that we have developed for today's computers will have to be revised or replaced to effectively operate at extreme scales" [ASCR 2009, p. 163].

In the past there have been several occurrences of disruptive changes in supercomputing hardware and concomitant changes in the programming interfaces, such as the change to vector processors and the rise of distributed memory computing with message passing (Fig 1.2).  In each case, a period of time was required for usage of the new programming techniques to mature within the HPC community.  This report describes some of the steps required to prepare for the next wave of change in hardware and programming environments and to decrease the period of time between formulation and maturation of new and effective programming techniques needed to exploit the new hardware.

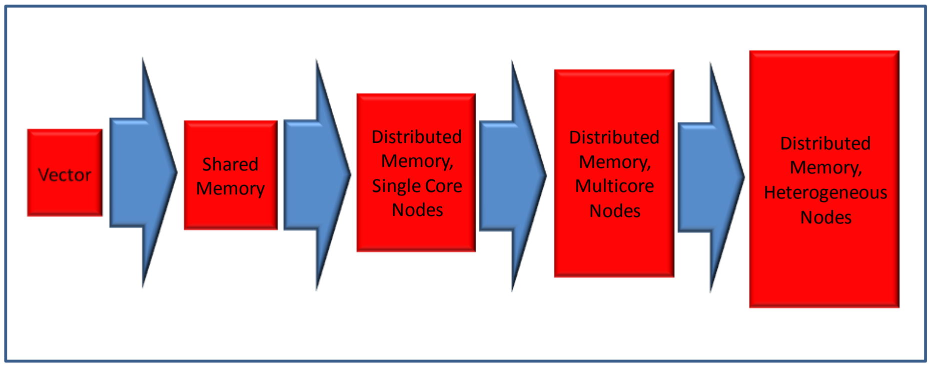

Vector
Shared Memory
Distributed Memory, Single Core Nodes
Distributed Memory, Multicore Nodes
Distributed Memory, Heterogeneous Nodes

**Fig. 1.2.  Historic shifts in HPC platform architectures**

## 1.3 Requirements Modeling

Requirements modeling is a formal process used to determine a path from the current state to a desired future state [Kothe and Kendall 2007]. Modeling the requirements for OLCF leadership computing involves the entire spectrum of elements involved in delivering breakthrough computational science, including science drivers and objectives, science models, computational algorithms, parallelization models, compilers, libraries and system software, computer hardware, software development processes, verification and validation procedures, application workflow, and data management. The requirements for these individual elements are interrelated because of the competing factors of science objectives and environmental limitations. For example, a science model requirement may dictate use of a certain algorithm, but available computer hardware may put limits on whether this algorithm can perform efficiently at scale.

For requirements to be useful, they must be actionable and as quantitative as possible. The OLCF requirements process is becoming increasingly quantitative as this process becomes better developed and more mature. The inherent nature of the leadership-scale computational science discovery process, however, makes it challenging at times to effectively manage requirements and limit risks [Bailey et al. 2007]. For example, disruptions in the hardware market can indirectly affect which software tools and development practices are needed. Unforeseen innovations in models or algorithms can eliminate some requirements or create new ones. Also, scientific discovery itself, which is not entirely predictable, can dynamically influence the set of requirements for accomplishing the next step in the scientific method. In addition, human factors such as HPC programmer productivity with various tools are not always well understood and sometimes can be determined only by experience. Despite these concerns, the OLCF requirements gathering and modeling process has identified clear needs for the next generation of leadership computing.

The first step of requirements development is data gathering and elicitation. The 2009 OLCF requirements effort utilized data from the following sources:

- Existing documentation regarding science objectives.
- Answers to science and application software questions from OLCF and Innovative and Novel Computational Impact on Theory and Experiment (INCITE) project proposal applications.
- A requirements survey to elicit project requirements from OLCF-supported projects (see Appendix B).
- In-depth interviews with science code project leaders and team members.

- Review of OLCF Leadership Computing usage logs over recent years to detect usage trends.
- Performance data generated on current computer hardware from key heavily-used OLCF applications.
- Survey of broader community and market trends.
- Discussions with computer hardware and software vendors regarding capabilities of next-generation offerings and related trends.

These data are then submitted to a requirements development process. The elements of this process include evaluating the data, detecting requirement commonalities across projects, evaluating tradeoffs, differentiating desired and required features, comparing findings against past experience and current and future resources, validating conclusions to minimize risks, and making recommendations. The requirements resulting from this process 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.

By using these requirements to manage and arbitrate decisions, the OLCF aligns leadership systems to the maximum possible extent with the needs and goals of the breakthrough science projects using these resources. This results in improved science quality and user productivity, higher fidelity physical models and numerical algorithms, more efficient and higher quality software, and better in-depth data analytics and workflow. The requirements modeling process also ensures that OLCF planning and procurement processes are in step with broader DOE and Office of Science goals. We expect that effective requirements development, management, and planning will positively influence the design, procurement, deployment, and operation of OLCF systems by measurably improving the quality, quantity and fidelity of the output of multiple breakthrough science simulation applications.

## 1.4 Organization of Report

To describe the motivating factors for next-generation computational science, Chapter 2 discusses the fundamental science goals driving the demands for leadership computing resources. Since computational science is rooted in application software, it is necessary to discuss the usage and resource requirements of leadership science applications. Thus, Chapter 3 discusses application usage and usage trends on OLCF systems, and Chapter 4 presents performance analysis of these codes. Chapter 5 discusses in detail the elements required to perform leadership science in the specified timeframe. Supplementary materials are presented in the appendices.

# 2. SCIENCE DRIVERS AND IMPACT

## *2.1 Science for the Nation*

High performance computing plays an increasingly strategic role in addressing urgent challenges in national and homeland security, energy security, economic competitiveness, health care, and environmental protection. The primary mission of the DOE ASCR program is to address these needs by discovering, developing, and deploying the computational and networking tools that enable researchers in the scientific disciplines to analyze, model, simulate, and predict complex phenomena important to the Department of Energy. This includes

- **Energy Security** – Computer simulation helps ensure America's energy security by enabling researchers to understand combustion, improve fuel cells, develop fusion energy and develop other technologies.
- **Nuclear Security** – ASCR-supported science contributes insights and research tools the National Nuclear Security Agency (NNSA) can use to ensure the safety and reliability of the nation's nuclear deterrent, a part of national strategy to safeguard America's nuclear security.
- **Scientific Discovery and Innovation** – ASCR hosts the most powerful open computing systems in the world, key to scientific discovery and economic competitiveness and leading to improvements in quality of life through innovation.
- **Environmental Responsibility** – Computer simulations help researchers understand mechanisms of environmental contamination and develop appropriate remediation technologies.

The Oak Ridge Leadership Computing Facility (OLCF) has made numerous contributions to advance these goals, including for example

- Discovery of new, critical phenomena in the deaths of massive stars and a new mechanism explaining the birth of pulsars;
- Breakthroughs toward understanding how turbulent flames stabilize within combustion devices, with implications for the design of gasoline engines, diesel engines and gas turbines;
- A fundamental new understanding of loss of thermal energy in tokamak fusion reactors, with ramifications for the design of the $10 billion ITER fusion device;

- Completion of new models of high-temperature superconductivity which bring us closer to designs for practical superconducting materials;
- Execution of the largest simulation ever of the dark matter cloud holding the galaxy together;
- Performing of one third of the climate simulation work for the most recent IPCC assessment.

Nonetheless, numerous science areas have identified the need for computational resources several orders of magnitude beyond what is currently available. Specific science goals are discussed below.

## 2.2 Science Drivers

### 2.2.1 Aerodynamics

| | |
|---|---|
| **DRIVER** | Need for accurate simulation of aerodynamic phenomena. |
| **STRATEGY** | Highly resolved simulation of fluid behavior using Large Eddy Simulation and Direct Numerical Simulation. This will require new software and methods to efficiently leverage high degrees of parallelism to deliver turnaround times useful for design. |
| **OBJECTIVE** | Create computing platform on which next generation software tools can be developed, validated, and applied, enabling groundbreaking aerodynamic advances. |
| **IMPACT** | Reduced fuel consumption and reduction in greenhouse gases throughout the economy. Improved flight safety. Enabler for a broad range of products and technologies that depend on fluid flows for efficient cooling, combustion, and robustness. |

Accurate simulations of aerodynamics and combusting flows are critical to our everyday life. These simulations are integral to the design of efficient transportation and power generation systems needed to reduce greenhouse gasses, reduce cost to businesses and individuals, and increase the country's energy independence. Fluid/structural interaction plays a key role in safety ranging from building design to medicine to air travel. However, there still remains a significant gap between actual physics and the computational models used to simulate regimes such as highly separated diffusing/accelerating flows. Today, phenomena such as turbulence and combustion chemistry are approximated in order to make them practical on existing computers. This sets up a "chicken and egg" situation – simulation software is designed for the computers of today, and the next generation of simulation software cannot be developed

**Geared Turbo Fan Nacelle System**

*Image courtesy: Pratt and Whitney*

and validated without next generation computing.

**OBJECTIVES FOR 20 PETAFLOPS:** Develop and apply simulation software for highly accurate simulation of complex flows across widely varying length and time scales, delivering results in a sufficiently timely manner to be used in product development.

**OBJECTIVES FOR EXASCALE:** Leverage petascale simulation tools in multipoint and multiphysics simulations.

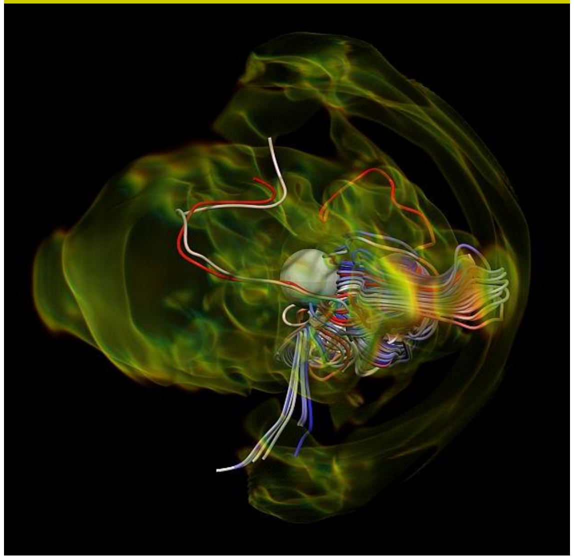
## 2.2.2 Astrophysics

| | |
|---|---|
| **DRIVER** | Core-collapse supernovae are among the universe's most prodigious explosions and produce most of the elements heavier than iron. |
| **STRATEGY** | Understand the death of massive stars and their contribution to galactic chemical evolution. |
| **OBJECTIVE** | Determination of the core-collapse supernova mechanism and the production of observational templates for a raft of observables: neutrino signatures, nucleosynthesis, gravitational waves, etc. |
| **IMPACT** | Predictive simulations of core-collapse supernovae will enable our fundamental understanding of the constituents of our world. |

Astrophysics research addresses physical phenomena from the smallest subatomic particles to the largest galaxies, such as the formation of elements, supernova behaviors, black holes, gravitational radiation, star formation and dark matter. Supernova occurrences are the most spectacular events in the universe and are fundamental to element formation. High-end scientific simulation can provide answers regarding how supernovae occur, what happens when black holes merge and what is the nature of dark matter.

**Fluid velocity streamlines during a Type II supernova collapse**

*Visualization by D. Pugmire, ORNL; Simulation by E. Endeve,C. Cardall, R. Budiardja, ORNL and UT–Knoxville, and A. Mezzacappa, ORNL.*



**OBJECTIVES FOR 20 PETAFLOPS:** Increase physical fidelity of nuclear burning module to effectively confront observations of SNe remnants and answer questions about galactic chemical evolution.

**OBJECTIVES FOR EXASCALE:** Determine the precise manner in which supernovae explode by incorporating quantum kinetics on macroscopic scales with realistic nuclear physics components to predict isotopic output.

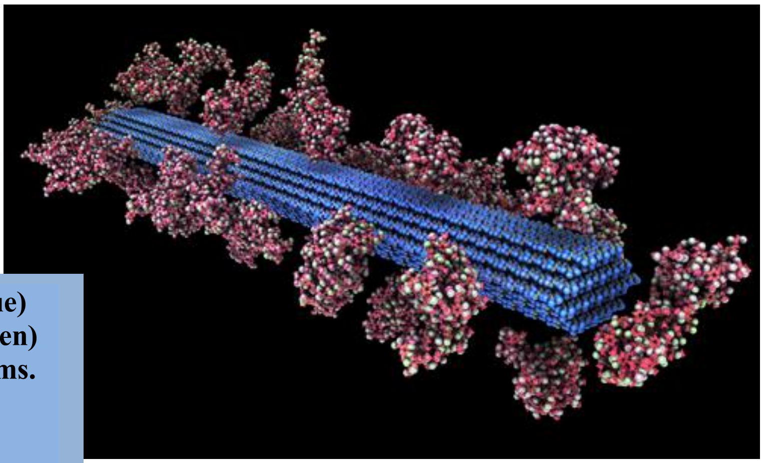
## 2.2.3 Biology

| | |
|---|---|
| **DRIVER** | Predict and simulate the behavior of complex microbial systems. Study mechanisms for converting cellulose to ethanol. |
| **STRATEGY** | Understand biomass cellulose recalcitrance and cellulose/enzyme interactions. |
| **OBJECTIVE** | Remove economic bottleneck in cellulosic ethanol production (recalcitrance to hydrolysis of biomass) by building realistic molecular simulation models encompassing both short and long length scales that will guide conceptual research in bioenergy. |
| **IMPACT** | Produce alternative to fossil fuels with drastically reduced environmental side effects. |

Microbial life affects nearly every physical and geochemical process on earth. HPC in biology brings the opportunity for design-driven research in areas such as biofuels, environmental remediation, protein research, climate change research and pharmaceuticals.

**OBJECTIVES FOR 20 PETAFLOPS:** Simulate microbial/biomass interface and dynamics of enzyme action on biomass. With multiscale development, simulate microbial enzymes acting on biomass. Incorporate more biomass component into model (lignin, hemicellulose, pectins, etc.).

**OBJECTIVES FOR EXASCALE:** Realistically simulate the properties of lignocellulose, requiring 40X increase in number of atoms and 10X increase in time scale.



**An atomistic model of cellulose (blue) surrounded by lignin molecules (green) comprising a total of 3.3 million atoms.**

*Image courtesy: Jeremy Smith and Jamison Daniel, ORNL.*

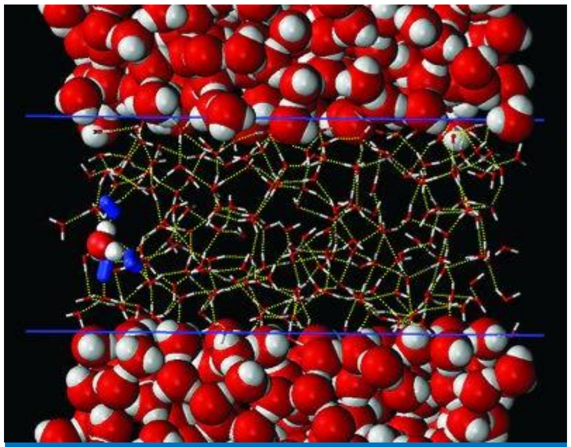
## 2.2.4 Chemistry

The use of electricity generated from intermittent, renewable sources requires efficient storage of this electrical energy. The performance of current electrical energy storage technologies falls far short of requirements for effective use. Basic research is critical to understand the fundamental electrochemical processes governing these devices.

**OBJECTIVES FOR 20 PETAFLOPS:** Larger length-scale required for accurate quantum mechanical description of electrochemical processes at interfaces; longer time-scale required to understand ion diffusion during charge-discharge cycles.

**OBJECTIVES FOR EXASCALE:** Solve larger systems to higher accuracy.

| | |
|---|---|
| **DRIVER** | Develop enhanced energy storage in nanostructured system. |
| **STRATEGY** | Study charge storage and transfer in nano-structured capacitors. |
| **OBJECTIVE** | Apply density functional theory to the workings of carbon tube supercapacitors—nanostructures that store two to three orders of magnitude more energy than conventional capacitors—and provide a nanoscale look at the physical/chemical processes that limit storage capacity, useful lifetime, and peak power output. |
| **IMPACT** | Revolutionize battery and other energy storage technologies. |



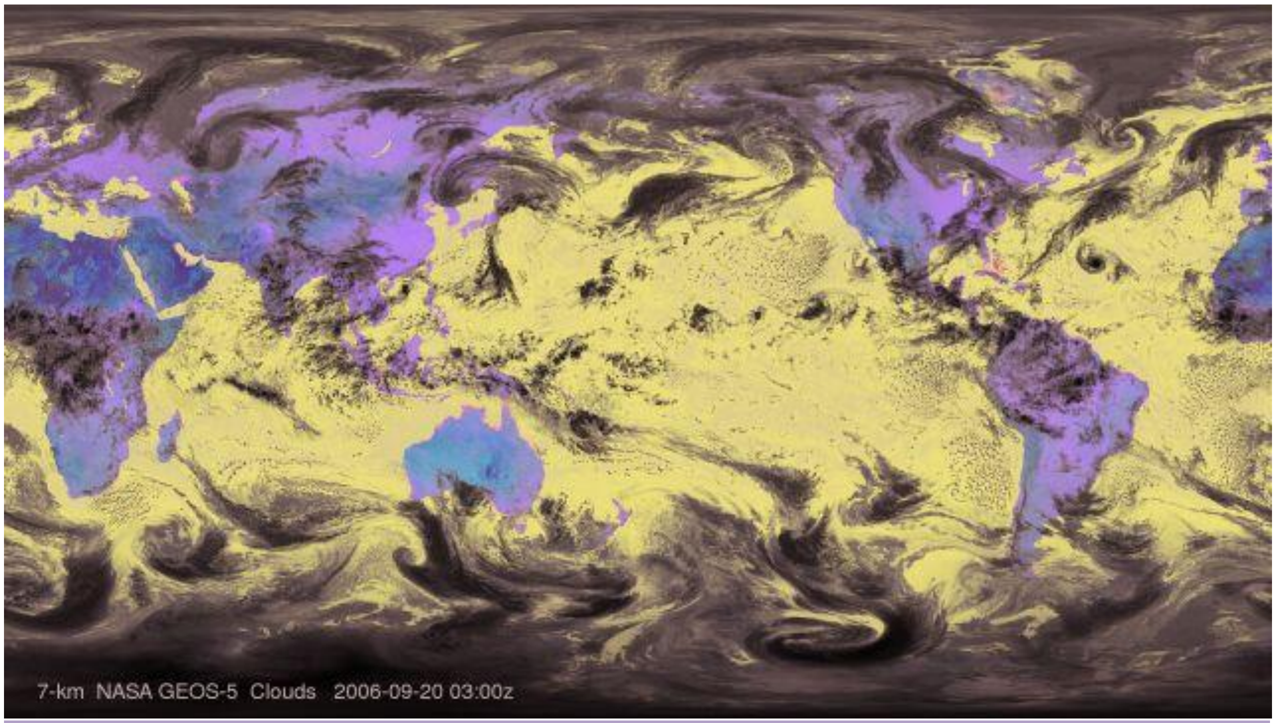
**$H_3O$+ undergoing chemical transformations at the air-water interface. The $H_3O$+ is depicted in the center using large red (oxygen) and three white (hydrogen) spheres. Blue bonds denote nearest hydrogen-bonded neighbors which are involved in interfacial chemistry.**

*Image courtesy: Chris Mundy, PNNL.*

## 2.2.5 Climate Modeling

| | |
|---|---|
| **DRIVER** | Accurately simulate scenarios of future climate change. |
| **STRATEGY** | Develop better models and higher resolution simulation capabilities. |
| **OBJECTIVE** | Improve the accuracy and fidelity of climate change predictions. Configure atmospheric, ocean, terrestrial, and cryospheric component models to answer policy and planning relevant questions about specific climate change adaptation and mitigation scenarios. |
| **IMPACT** | Mitigate deleterious effects of global climate change. |

Concerns regarding global warming and anthropogenic climate change drive the need to improve the scientific basis for assessing the potential ecological, economic and social impacts of climate change. More accurate climate models can simulate different scenarios of possible future climate change to help policy makers in their planning processes.



**A global snapshot of forecasted cloud cover from the GEOS-5 model at 7-km global resolution, highlighting the existence of numerous cloud types represented at these resolutions, including hurricane Helene in the tropical Atlantic Ocean, convective clusters throughout the tropical Pacific and Indian oceans, and the marine stratocumulus layer off the west coasts of North and South America, along with the characteristic structures of mid-latitude cyclones forming along fronts in both the Northern and Southern hemispheres.**

*Image courtesy: Max J. Suarez, NASA.*

**OBJECTIVES FOR 20 PETAFLOPS:** Create higher fidelity simulations with improved predictive capability on decadal time scales using high-resolution Earth System Model configurations. Realistically represent features like precipitation patterns/statistics and tropical storms.

**OBJECTIVES FOR EXASCALE:** Develop higher resolution models to support regional climate modeling. Improve modeling of physical, chemical and biological processes. Simulate the carbon cycle. Explore parameters giving rise to uncertainties.

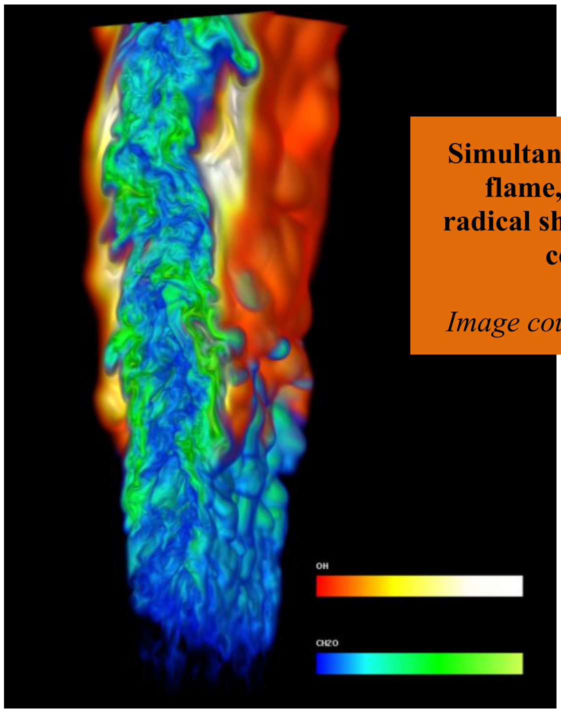
## 2.2.6 Combustion

| | |
|---|---|
| **DRIVER** | Optimize the design of lean, premixed turbine combustors. Simulate advanced engine concepts under different operating conditions. Evaluate combustion behavior of new biofuels. Improve thermal efficiency by potentially 25%-50%. |
| **STRATEGY** | Use direct numerical simulation for fundamental studies of the microphysics of turbulent reacting flows. |
| **OBJECTIVE** | Increase clarity of modeling of mixing transition. Model biofuel surrogates to provide insight into next generation engines. Model lifted flame stabilization, extinction and reignition, premixed flame structure. |
| **IMPACT** | Clean, efficient burning processes to supply energy needs. |

Combustion currently supplies 85% of America's energy needs. Environmental, economic and national security concerns are driving a shift toward alternative fuels for combustion. These new fuels have different physical and chemical properties leading to different combustion processes. Advanced simulation will determine effective designs for combustion systems, to make cleaner, more efficient use of combustible fuels.

**OBJECTIVES FOR 20 PETAFLOPS:** Attempt 50–100% increase in Reynolds number (from 10K to 15–20K) with existing chemical complexity (single-stage ignition ethylene with 22 species) at ambient pressure; explore fully developed turbulence beyond theoretically predicted "mixing transition". Attempt more biofuel-like chemistry while keeping complexity constant, e.g. going to dimethyl-ether. Alternatively stay at high pressure (50 atm) with more complex chemistry (iso-butanol: practical biofuel), namely 60–80 species.

**OBJECTIVES FOR EXASCALE:** Develop fundamental understanding of "turbulence-chemistry" interactions of non-petroleum based biofuels at high pressure engine conditions. Accurately model lifted flame stabilization in ignitive flows, low-temperature ignition kinetics coupling with transport, extinction/reignition in dilute heated mixtures, emissions and soot, controlling inhomogeneous autoignition for HCCI combustion by tailoring mixing and ignition kinetics, increased turbulent Reynolds numbers.



**Simultaneous volume rendering of a lifted ethylene/air slot jet flame, where the lifted flame is represented by hydroxyl radical showing the flame stabilization point. The particles are colored by temperature: cold (blue), hot (red).**

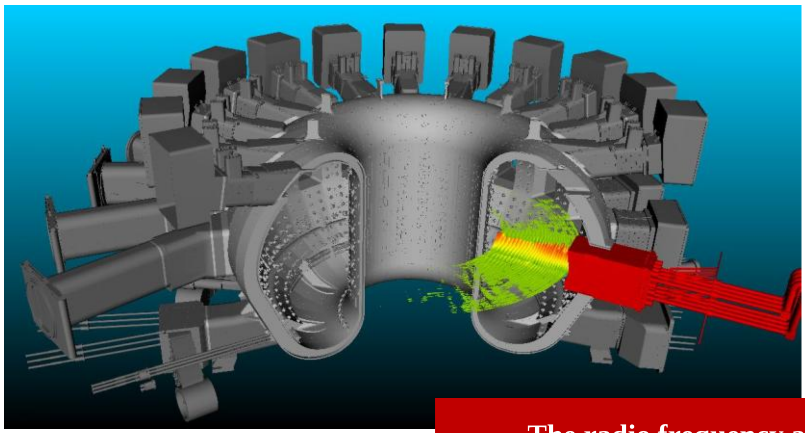
*Image courtesy: Jackie Chen, SNL, and Kwan-Liu Ma, UC Davis.*

## 2.2.7 Fusion

| | |
|---|---|
| **DRIVER** | Effectively model and control the flow of plasma and energy transport in a fusion reactor. |
| **STRATEGY** | Understand "hot spots" near antenna surface, "parasitic" draining of heat to the plasma surface in small reactors. |
| **OBJECTIVE** | Use first principle simulations of the gyrokinetic equations, to study cascades and propagation in Collisionless Trapped Electron Mode (CTEM) turbulence, as well as to study the electron temperature and Ion temperature gradient (ITG/ETG) drift turbulence for the ITER reactor. Include the full spectrum of toroidal harmonics for specific antenna geometries. Study propagation and absorption of lower hybrid waves. |
| **IMPACT** | Clean, nearly limitless energy supply. |

Fusion energy offers the potential for a source of clean, virtually unlimited power. Successful fusion energy depends on the ability to heat and electromagnetically confine the reactive plasma within the fusion reactor for a sufficient period of time. Advanced computational simulation of this plasma is essential to designing a working fusion reactor. Currently the primary focus is on the ITER reactor, scheduled for deployment in 2018.

**OBJECTIVES FOR 20 PETAFLOPS:** Scaling to realistic Reynolds numbers needed to address burning plasmas. Include multi-scale integrated electromagnetic turbulence in the whole-volume ITER plasma in realistic diverted geometry. Model plasma edge and core turbulence.



**The radio frequency antenna (red) launches three-dimensional wave fields into the ITER plasma. The waves heat deuterium and tritium fuel to fusion temperatures about ten times hotter than the surface of the Sun.**

*Image courtesy: Fred Jaeger and Sean Ahern/U.S. ITER Project Office.*

**OBJECTIVES FOR EXASCALE:**

Perform integrated modeling of the entire discharge cycle of magnetically confined fusion plasmas. Increase spatial resolution by an order of magnitude. Model longer energy confinement time scales. Develop new robust validated numerical algorithms to support these calculations.

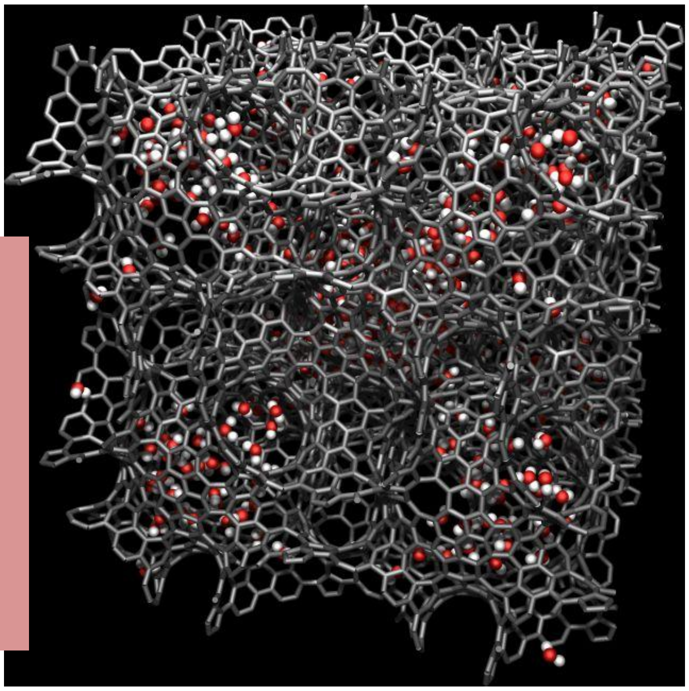
### 2.2.8 Materials Science

| | |
|---|---|
| **DRIVER** | Design high temperature superconductors for improved energy transmission and oxide electronics. |
| **STRATEGY** | Hone in on the theory for high temperature superconductors. Apply Humbbard model to understand role of inhomogeneities. |
| **OBJECTIVE** | Understand the quantitative differences in the transition temperatures of high temperature superconductors. Develop a more complete understanding of the pairing mechanism in cuprates, including the role of chemical composition, disorder, and nano-scale inhomogeneities. Develop a more complete understanding of the pairing mechanism in cuprates, including the role of chemical composition, disorder, and nano-scale inhomogeneities. |
| **IMPACT** | Nanoscience and nanotechnology capabilities to increase US competitiveness and industrial leadership. |

Materials science is an interdisciplinary field that incorporates chemistry, physics, and engineering to provide a deeper understanding of existing materials and to allow for the design of new materials with predetermined properties. Research into the nature of materials promises to revolutionize many areas of modern life, from power generation and transmission to transportation to the production of faster, smaller, more versatile computers and storage devices. High performance supercomputing is strategic to understanding the phenomenon of high temperature superconductivity, with potential application to energy transmission and other areas.

**OBJECTIVES FOR 20 PETAFLOPS:** Model magnetic/superconducting phase diagrams including effects of disorder. Model the effect of impurity configurations on pairing and the high-T superconducting gap. Model the high-T superconducting transition temperature materials dependence in cuprates.

**OBJECTIVES FOR EXASCALE:** Increase accuracy and fidelity of superconductivity simulations.



**Molecular dynamics simulation of confinement and dispersion of small molecules within carbon nanostructures, mimicking the dynamics of electrolytes in porous carbon materials.**

*Visualization by the SciDAC code VisIt. Simulation by Dr. Vincent Meunier, ORNL; and visualization by Jeremy Meredith and Sean Ahern, ORNL.*

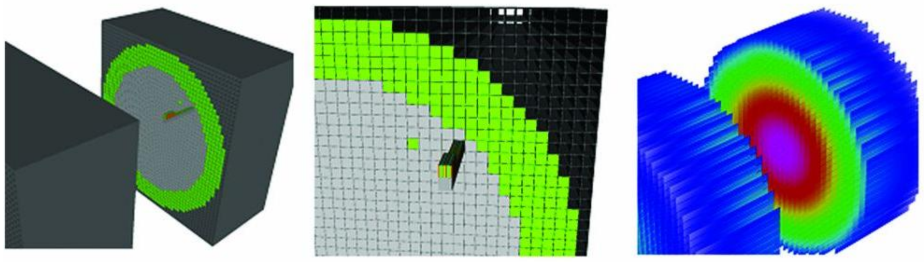
## 2.2.9 Nuclear Energy

| | |
|---|---|
| **DRIVER** | Effectively model nuclear reactor behavior as a part of national energy security strategy. |
| **STRATEGY** | Predictive simulation for reactor core and facility shielding. |
| **OBJECTIVE** | Implement mathematically consistent algorithms for multiscale modeling of radiation transport in the core of the nuclear reactor. |
| **IMPACT** | Better, safer reactor design with increased availability of low-cost energy. |

Over the last several years, the energy security of the United States has risen in importance both politically and economically. Our nation needs to increase energy security, reduce dependence on unreliable sources of energy, obtain energy at affordable prices, and insure that the environment is not impacted. Improving scientific understanding of the behaviors of nuclear fuels, reactors, separation processes and long-term waste management sites will increase the viability of nuclear energy strategies for addressing these concerns.

**OBJECTIVES FOR 20 PETAFLOPS:** Predict behavior of existing and novel nuclear fuels and reactors in transient and nominal operation and evaluate predictability of software through uncertainty quantification and sensitivity analysis. Model full-core reactor neutronics, neutronics/hydraulics coupling, accident scenarios, fast reactor transients. Increase fidelity of solutions by replacing existing homogenization techniques with direct techniques and solving for time-dependent multiphysics.

**OBJECTIVES FOR EXASCALE:** Develop integrated performance and safety codes with improved uncertainty quantification and bridging of time and length scales. Implement next-generation multi-physics multiscale models. Perform accurate full reactor core calculations with 40,000 fuel pins and 100 axial regions. Perform ultra-resolution simulations for convergence verification.



**Two pictures (left and center) of Zero Power Reactor Experiment 6/6A geometry and uranium-235 plate power distribution (with separated matrix halves). The gray indicates the matrix tube and drawer fronts that are loaded into each tube position. The solid green squares are 2-inch depleted uranium metal blocks directly loaded into the main core and acting as a neutron blanket. The plot at the right shows the enriched uranium plate power with the matrix halves separated.**

*Image courtesy: Dinesh Kaushik, ANL.*

### 2.2.10 Nuclear Physics

| | |
|---|---|
| **DRIVER** | Achieve a consistent theoretical formulation to accurately describe and predict nuclei and their reactions. |
| **STRATEGY** | Develop a predictive microscopic nuclear theory grounded in the fundamental interactions between protons and neutrons, including higher-body forces. |
| **OBJECTIVE** | Use ab initio techniques to solve the nuclear many-body problem using QCD (quantum chromodynamics) derived 2-, 3, and higher-body interactions and density functional theory for studying light, medium, and heavy mass nuclei. |
| **IMPACT** | Fundamental understanding of the nuclear force and predictive capability for exotic nuclei unable to be produced in a laboratory to drive future experimental facilities/programs and nuclear energy programs. |

A detailed understanding of the atomic nucleus is both fundamentally important and of great practical significance. Not only important to explaining the birth of the universe and astrophysical phenomena, understanding nuclei is crucial in energy generation as well as in industrial and medical applications. Nuclear physics focuses on predicting and explaining rich classes of phenomena that occur in nuclei. The theoretical goal of increased predictive power for nuclear processes that occur in nature or in nuclear reactors, but cannot be measured in the laboratory with sufficient precision, drives the field to achieve detailed simulations using extreme-scale computers and cutting-edge algorithms.

**OBJECTIVES FOR 20 PETAFLOPS:** High precision ab initio calculations for light ion reactions.

**OBJECTIVES FOR EXASCALE:** Nuclear fission calculations using ab initio techniques.

### 2.2.11 Thermoelectric Materials Science

| | |
|---|---|
| **DRIVER** | Improve vehicle fuel efficiency. |
| **STRATEGY** | Understand and predict atomic arrangements and growth mechanisms of nano-structured bulk thermoelectric materials, which show significant promise to achieve higher efficiency. |
| **OBJECTIVE** | Establish unambiguous atomically resolved structural assignment for the nanocomposite and identify the mechanisms for the nucleation and atomic arrangement of the nanoprecipitates, which ultimately determines the thermoelectric properties of nano-structured materials. |
| **IMPACT** | Convert vehicle exhaust heat into electricity in an effort to reduce fossil fuel consumption. |

Waste heat is one of our most abundant sources of alternative energy. Thermoelectric energy conversion offers the opportunity of converting waste heat into useful electricity to achieve improved energy efficiency. HPC in thermoelectric materials brings the opportunity for design-driven research in areas such as atomic-level materials understanding and optimization.

**OBJECTIVES FOR 20 PETAFLOPS:** The density functional theory (DFT) calculations employ a massive supercell containing ~ 2000 atoms which can accommodate the composite system with the size of the fully enclosed nanoprecipitate reaching those observed in actual specimens.

**OBJECTIVES FOR EXASCALE:** Realistically simulate the properties of real materials, requiring 40X increase in number of atoms and 10X increase in time scale.

### 2.2.12 Turbomachinery

| | |
|---|---|
| **DRIVER** | Design jet engines, gas turbines, and steam turbines with increased efficiency (less fuel burn) and increased durability. |
| **STRATEGY** | Use supercomputers to assess fundamental questions that would allow the development of new technology used to impact future designs. |
| **OBJECTIVE** | Some of the fundamental questions that need to be answered include (but are not limited to): (a) understanding multistage interactions in compressors, (b) understanding aero-thermal effects in high pressure turbines, and (c) understanding turbulent physics in low pressure turbines. |
| **IMPACT** | Turbines for more efficient jet engines and electrical generators. |

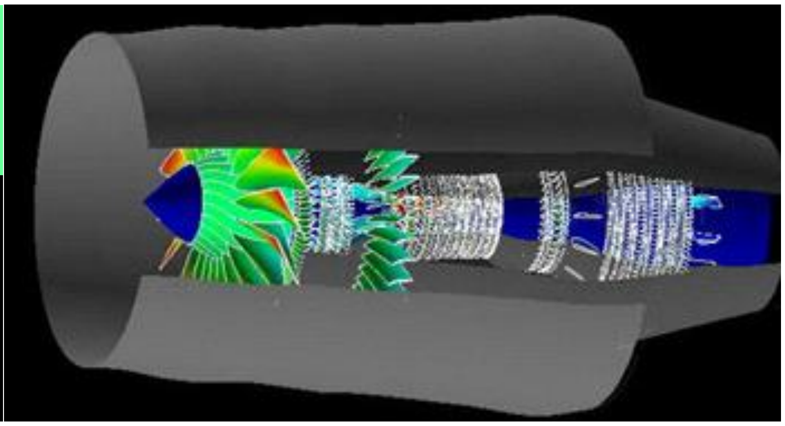
Aviation propulsion and power generation depend on rotating turbomachinery, e.g. jet engines and gas turbines. Modern day designs are developed using CFD analyses that can be run on conventional gigaflop clusters. As we continue to push the state of the art to improve efficiency, the problem sizes (grid density and choice of computational domain) are growing and the complexity of the physical models are increasing. These twin effects drive us towards petaflop and beyond computations in support of future designs.

**OBJECTIVES FOR 20 PETAFLOPS:** Enable the use of CFD on full components, e.g. CFD of the entire high pressure compressor, using basic turbulence models.

**OBJECTIVES FOR EXASCALE:** Enable the use of CFD on a full component using 1st principle turbulence models (DNS or LES); enable full engine simulations.

**Simulation of GE Turbofan Engine**

*Image courtesy: NASA*



# 3. SCIENCE APPLICATION USAGE

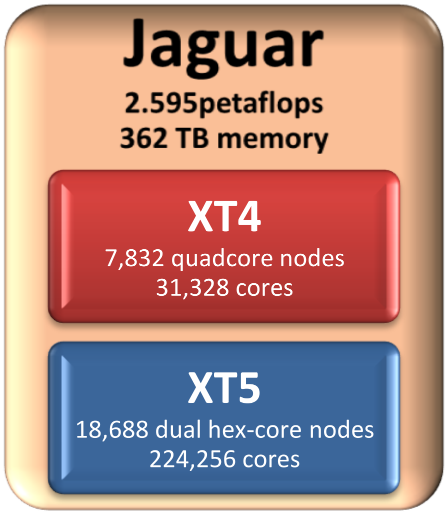
Ongoing assessment of the need for and usability of leadership-class computing resources is crucial to its future direction. This raises the questions: "What is leadership-class computing?" and "What metrics determine need and usability?" Leadership-class computing has long been synonymous with *capability computing* described by Graham et al. [Graham et al. 2005]. Capability computing is defined as, "using the maximum computing power to solve a large problem in the shortest amount of time," thus solving "a problem of a size or complexity that no other computer can." This is in contrast with *capacity computing* which uses, "efficient cost-effective computing power to solve somewhat large problems or many small problems." When discussing capability systems, peak flops is often the performance measure of computing power, evident in the popularity of the TOP500 list [TOP500]. On the other hand, experience in real scientific computing shows that the actual measure of computing power also includes other system attributes (e.g. number of CPU-hours deliverable, random access memory, memory bandwidth, interconnect bandwidth, etc.).

In connection with these definitions, metrics of leadership-class computing include scaling to larger numbers of processors to tackle larger problem sizes and complexities and reduce times-to-solution. Application scaling is a good indicator of need for leadership-class computing resources. Obviously there is apparent need when applications require a large fraction of the resource for a single calculation. Applications scale to larger numbers of processors to utilize more flops and, as is more often the case, to distribute the memory load of data-intensive calculations. Without distinguishing between flops or memory as the primary motivation for scaling to higher core counts, we track maximum job sizes, as well as utilization by job size of our user community. We bin this usage data into three job size categories: usage of less than 20%, between 20 and 60%, and greater than 60% of the computing resource. The typical scale of a code appropriate for a leadership-class system is utilization of greater than 20% of the resource for a single calculation. Job sizes less than 20% of the resource can typically fit onto smaller, capacity systems.

Time-to-solution is more difficult to map to a leadership computing metric, since the relationship of computing resource size to the application's scaling behavior and science discovery workflow characteristics may be complex. Generally, the assumption is a larger system with a higher peak flop count provides a faster time-to-solution. But, more than peak flops, the sheer quantity of CPU-hours available and consumed imply the likelihood of a faster time-to-solution. A leadership-class computing resource typically delivers a higher number of CPU-hours. Without these hours, science would not be accomplished at a rapid pace. As a broad metric, we can measure science time-to-solution through CPUhour utilization. Utilization of CPU-hours is also an indicator for usability of the system across the breadth of disciplines capable of utilizing the resource. Since supercomputing is a scientific tool that transcends any one discipline, its overall effectiveness is tied with its utility to a multitude of disciplines.

To better understand the impact of OLCF leadership-class computing resources on computational science efforts, we present here usage statistics as an unbiased and quantitative measure of both need and usability. We can also make projections based on usage patterns regarding the likelihood and speed of adoption of new computing resources inevitable on the path toward exascale computing. This chapter highlights usage trends on the OLCF Jaguar supercomputer, separately for the XT4 and XT5 partitions.

The Jaguar supercomputer provides the largest fraction of computing time by the OLCF. It currently provides users with an aggregate peak performance of 2.595 petaflops, 362 terabytes of system memory, 10.7 petabytes of disk space, and 240 gigabytes/second of disk bandwidth. There are two partitions of Jaguar: the XT4 partition contains 7,832 compute nodes with quadcore AMD Opteron 1354 (Budapest) processors, totaling 31,328 processing cores, and the XT5 partition, with 18,688 compute nodes with dual hex-core AMD Opteron 2435 (Istanbul) processors, totaling 224,256 processing cores.



The XT5 partition became available to our larger user community in July 2009; for the first half of 2009, during its transition-to-operations period, it was only open to select Early Science users. The Early Science period was prior to the hex-core upgrade of Jaguar. At that time, the XT5 partition had 18,688 compute nodes with dual quadcore AMD Opteron 2356 (Barcelona) processors, totaling 149,504 processing cores. Early Science utilization statistics provided in this section reflect usage of the XT5 prior to the hex-core upgrade. The Early Science users were composed of high-end users with pertinent science need and with applications that typically scaled to greater than 20% (~ 30,000 cores) of the system resource.

## *3.1 User Demographics*

The wide reach of high-performance computing across many science domains is a testament to its usefulness as a scientific tool and a measure of its usability. The user population tends to be diverse and at various programming proficiency levels which creates a challenge for user support. The multi-tiered and integrated support structure offered by the OLCF as described in Appendix C has been an optimal approach for assisting and understanding our user population. Understanding user needs and the extent that each science domain actively tries to benefit from high-performance computing will likely guide tools development on emerging platforms. Though a standard set of libraries and software tools benefit the user community at large, additional domain-specific tools for increased user productivity are a possibility on more complicated systems. The goal is to ensure high resource usability to support a high level of user productivity and science output as computing shifts to achieve exascale.

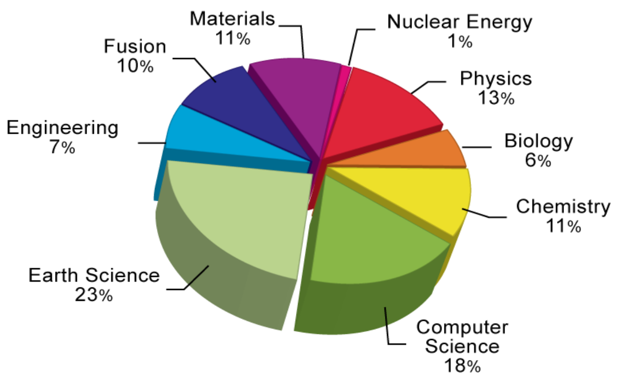


**Fig. 3.2. OLCF 2009 total user demographics by science category**

**Table 3.1. Research areas and science domains by INCITE categorization**

| Science Category | Represented Research Areas |
|---|---|
| **Biology** | Bioinformatics<br>Biophysics<br>Life Sciences<br>Medical Science<br>Neuroscience<br>Proteomics<br>Systems Biology |
| **Chemistry** | Chemistry<br>Physical Chemistry |
| **Computer Science** | Computer Science |
| **Earth Science** | Climate<br>Geosciences |
| **Engineering** | Aerodynamics<br>Bioenergy<br>Combustion<br>Turbulence |
| **Fusion** | Fusion Energy<br>(Plasma Physics) |
| **Materials** | Materials Science<br>Nanoelectronics<br>Nanomechanics<br>Nanophotonics<br>Nanoscience |
| **Nuclear Energy** | Nuclear Fission<br>Nuclear Fuel Cycle |
| **Physics** | Accelerator Physics<br>Astrophysics<br>Atomic/Molecular<br>Physics Condensed<br>Matter Physics<br>High Energy Physics<br>Lattice Gauge Theory<br>Nuclear Physics<br>Solar/Space Physics |

The majority of the hours ( > 75%) delivered by the OLCF are utilized by INCITE (Innovative and Novel Computational Impact on Theory and Experiment) projects. The INCITE program awards sizeable allocations (on the order of millions of processor-hours per project) for large-scale, computationally intensive research projects at America's premier leadership computing facilities (LCF), established and operated by the U.S. Department of Energy (DOE) Office of Science. These INCITE projects address grand challenges in science and engineering, such as developing new energy solutions and gaining a better understanding of anthropogenic climate change. Table 3.1 shows the science domains and research areas requesting and utilizing INCITE allocations at the OLCF.  Utilization of high-performance computing resources touches every field of science and is a valuable tool in scientific discovery.

The roughly 800 active users on the Jaguar supercomputer utilize approximately 70 different applications.  A profile on applications can be found in Appendix D.  Fig. 3.1 shows the distribution of users in the complete OLCF user community by their represented science category.  This includes users from INCITE and Director's Discretion projects and internal developers.  The largest number of users is from the earth sciences community, which is largely comprised of climate research, accounting for 23% of the user population.  The next largest group of users is from Computer Science, which includes a large number of internal developers who maintain and test system resources and create new tools and optimization strategies.  Mid-size communities utilizing the system include user communities in materials science, chemistry, fusion/nuclear energy, and physics.  The smallest numbers of users come from engineering and biology.  Although INCITE project principal investigators typically have a high degree of HPC proficiency, the users of a project span a wide range of skill levels.  Science users range from end-users that run an established application and analyze the output to developers who actively improve and scale their applications for frontier science breakthroughs.

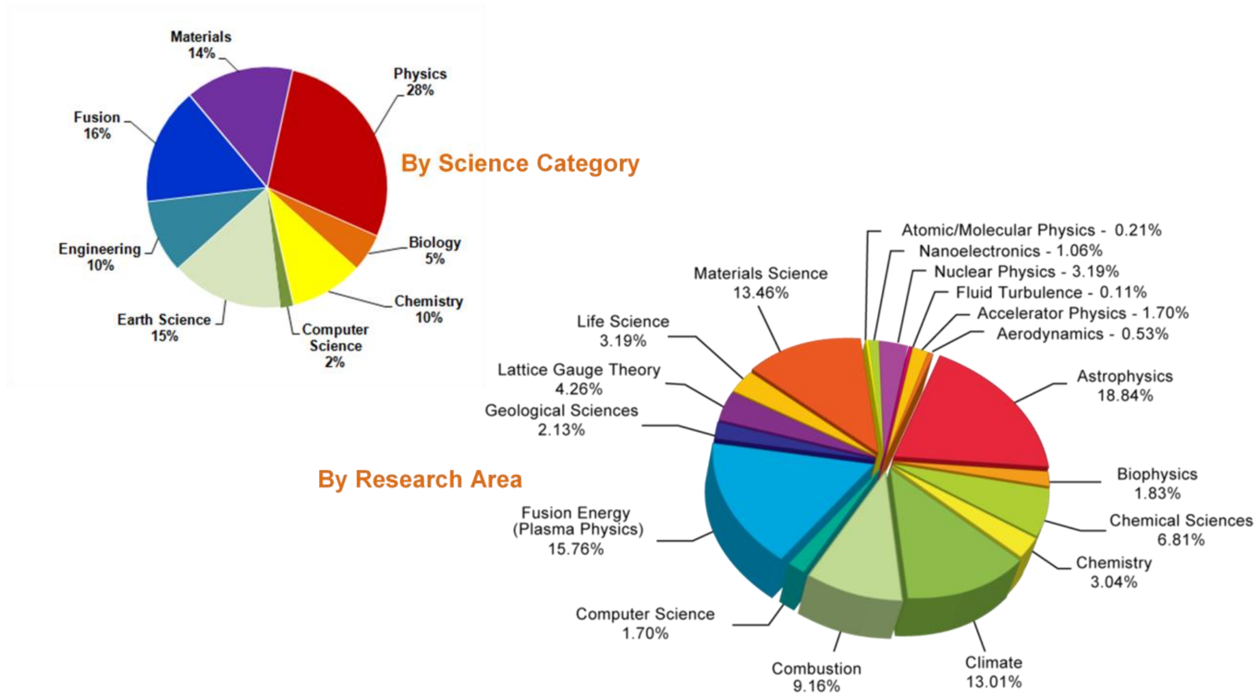


**Fig. 3.3. INCITE 2009 allocations by science category and research area**

Interestingly, the number of users does not necessarily correlate with usage. When comparing the user distribution of Fig. 3.1 and the INCITE allocations awarded for 2009 by science category in the top left pie chart in Fig. 3.2, we see that the largest usage is not from the largest user communities of earth science and computer science, but rather from physics. Further breakdown of the allocations by research area in Fig. 3.2 shows that astrophysics, an early adopter of high-performance computing, dominates physics usage, accounting for nearly 70% of the allocation. As early adopters of HPC, astrophysics applications are more mature and capable of utilizing larger resources on a single run. These applications typically are computationally intensive, accounting for multiple physics phenomena. Conversely, the lower computing time utilized by the earth sciences with the higher number of users suggests their applications require further development, and have yet to achieve maximal scaling for the resource.

The disparity between the high number of computer science users and very low INCITE allocation is due to the high number of internal staff developers falling into the category of computer science. The computing time for internal developers does not fall under the INCITE projects. The utilization from the other science categories of fusion, materials, chemistry, biology, and engineering is consistent with the size of their user base. The wide representation of science domains actively using Jaguar implies a high level of usability of the system, despite the varied skills of the user population.

## *3.2 Usage Statistics*

### 3.2.1 Overall Usage

Allocations and usage of OLCF computing resources have grown steadily over the last few years as shown in Fig. 3.3. This graph shows that user core-hour consumption increases as more core-hours become available from year to year. The CPU-hours requested and used are consistent, showing that the computing resources are not wasted. As computational science continues to grow into an invaluable method of scientific investigation across all science domains, demand for larger computing resources and more computing time will also increase.

Fig. 3.4 shows the number of INCITE projects on the OLCF systems between the years 2006 and 2009. Although Fig. 3.3 shows dramatic increases in the number of CPU-hours utilized from year to year, Fig. 3.4 shows that this is due only to a modest increase in the number of projects, with some science categories remaining constant in the number of projects from year to year. This indicates that applications are maturing from year to year and are capable of utilizing more of the available resource, and that the science projects are becoming more computationally intensive. As more applications improve their scaling capabilities, the number of CPU-hours available needs to increase. Additionally, to accommodate new projects from year to year, a substantial increase in available computing time needs to be addressed. This implies the need for larger and/or more leadership-class computing resources.

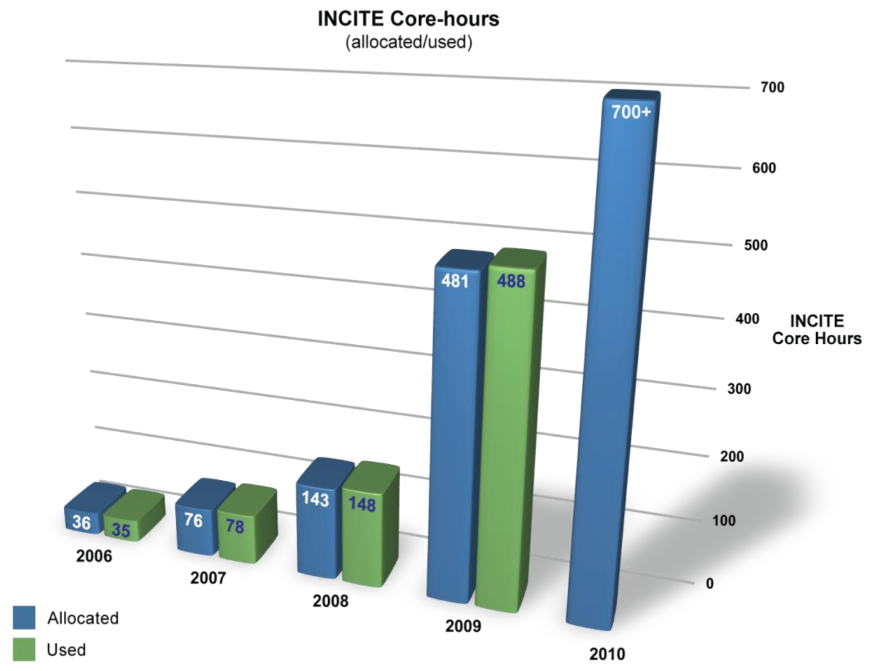


Fig. 3.4. Allocated and used INCITE core-hours (CY 2006 - 2010)

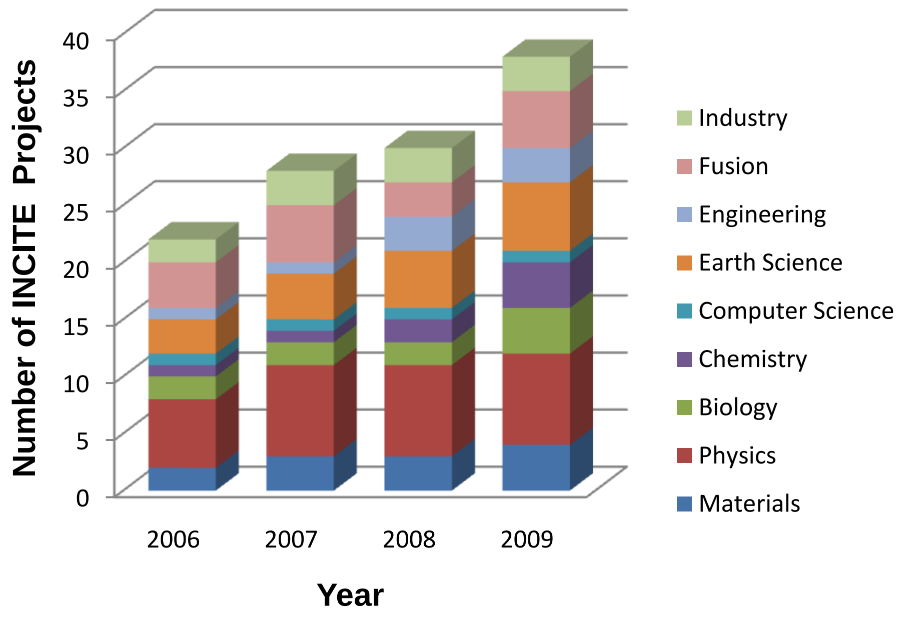


Fig. 3.5. Number of INCITE projects by science category (CY 2006 - 2010)

### 3.2.2 Job Size Distribution

Fig. 3.5 shows the utilization of Jaguar broken down by job size distribution for INCITE project years 2008 and 2009 on the XT4 partition and for 2009 Early Science projects on the XT5 partition. Job size

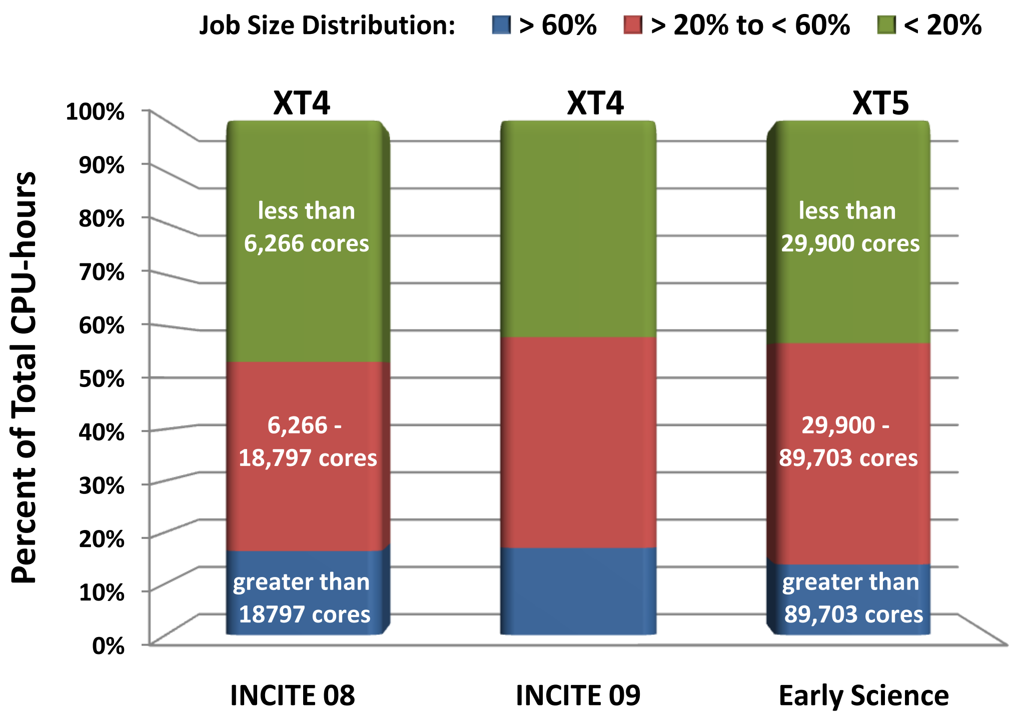


**Fig. 3.6. Utilization of Jaguar by job size for INCITE projects (CY 2008 - 2009) and Early Science projects**

is defined as the number of cores utilized during any given calculation. The exact number of cores for each job size is shown in Fig. 3.5. The percentage of utilization for a job distribution range is defined as the total number of core-hours utilized in that job size range divided by the total number of core-hours used on that particular system over the period of interest. Regardless of the calendar year, system or project type, the OLCF utilization has a job distribution load characteristic of a leadership (or capability) usage model, namely, skewed heavily toward usage of a large percentage (i.e. >20%) of the total available resource for any given calculation. Greater than 50% of the utilization by OLCF users requires more than 20% of the resource for a single calculation. Surprisingly, the same job size distribution is seen on the XT5 partition as on the XT4, despite having five times as many cores. The ability of the user population to scale their applications quickly for new resources is a promising sign for the adoption of new platforms. Likely, the similarity in architecture and manner in which users access and utilize the system resources eased the transition from the XT4 to the XT5.

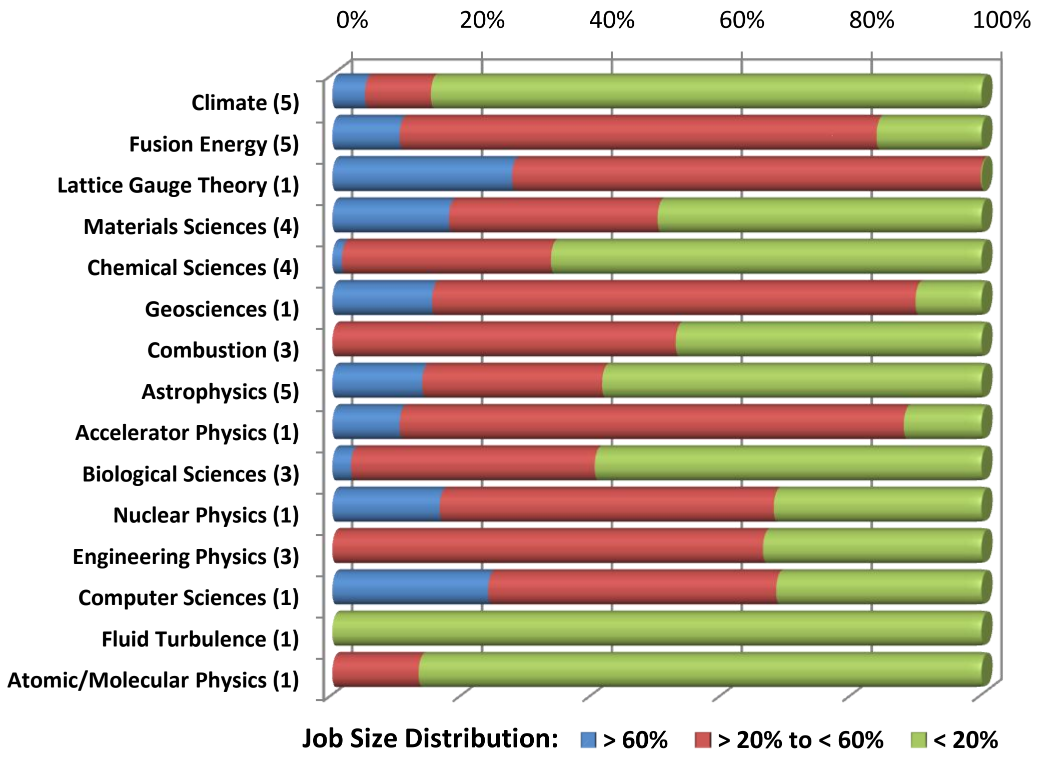


**Fig. 3.7. Utilization of Jaguar XT4 by job size for 2009 INCITE projects by research area**

For the 2009 INCITE projects on the XT4 partition, Fig. 3.6 shows an additional break down of the current utilization data with job size distributions for each research area. Research areas are listed along the *y*-axis and ordered by utilization, with the largest usage at the top (climate) and the value in parentheses being the number of projects contributing to the statistics. The majority of research areas are utilizing the XT4 resource appropriately for capability computing, with the greater part of their jobs using more than 20% of the resource. Lattice gauge theory, geosciences, accelerator physics, and fusion energy are particularly strong HPC research areas, capable of using more than 20% of the machine for more than 80% of their usage. Research areas that are below capability utilization include fluid turbulence and atomic/molecular physics. Climate research, though typically using less than 20% of the resource on any single run, utilizes a large number of CPU-hours. Although climate researchers could run single simulations on a smaller capacity system, typically one study requires a large aggregate of runs with different parameters. To address time-to-solution, climate research consumes a large number of CPU-hours only deliverable by leadership-class computing resources, but at a smaller scale. Fusion energy and lattice gauge theory applications not only utilize a large number of CPU-hours but exhibit leadership-class scaling. Applications in these research areas are potentially good candidates as early adopters of new platforms due to the maturity of their codes and application teams.

The increasing competition for time on leadership-class systems will most likely alter the usage distribution. Applications that scale to exploit the benefits of the size and speed of leadership-class resources will take priority over those that simply need more CPU-hours. The user community will need to make an active effort to continue developing their applications. The importance of application development as a crucial component of the scientific investigation cycle will become more evident with changes to the computing framework for exascale.

## 3.2.3 Petascale Early Science Usage

The Early Science utilization of the XT5 partition during the transition-to-operations period over the first half of 2009 requires special attention. Select users were given access to the XT5 petascale system with 149,504 cores. The largest resource available to these users previously was the Jaguar XT4 partition, with 31,328 cores. Applications scaling to 20% of the XT5 system meant that they were using 100% of the XT4 partition for similarly-sized jobs. Thus, it is quite impressive that many applications were able to so quickly utilize more than 20% of the resource for a single calculation. The speed at which the user community was able to develop their codes at this tremendous scale is remarkable.

Fig. 3.7 shows the Early Science utilization of the XT5 for each science category. All science domains except computer science are represented, and fusion and nuclear energy are combined in this figure. The largest utilization of the resource is from materials science at 25%, with chemistry (20%) and fusion (17%) close behind. Physics, biology, engineering, and earth sciences utilized around 10%.

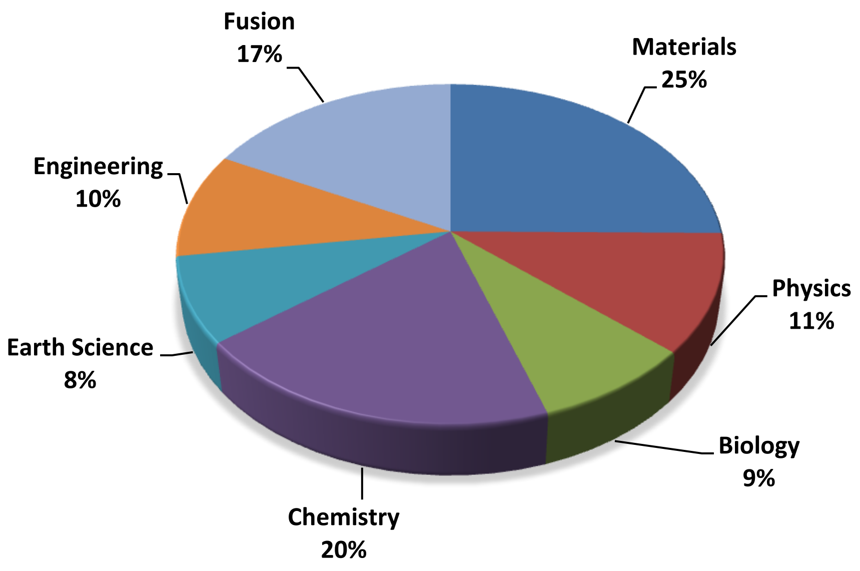


**Fig. 3.8. Early Science usage by science category**

**Table 3.2. Early Science projects on Jaguar XT5**

| Project ID | Allocation (CPU-hours) | Principal Investigator | Project Description |
|---|---|---|---|
| AST017 | 57,000,000 | Tony Mezzacappa | A Three-Dimensional Model of SN1987A |
| AST018 | 57,000,000 | James Van Meter | Frontier Simulations of Black Hole Mergers |
| MAT014 | 57,000,000 | Thomas Schulthess | Investigations of the Hubbard Model with Disorder |
| CMB006 | 50,000,000 | Jacqueline Chen | Direct Numerical Simulation of Diesel Jet Flame Stabilization at High Pressure |
| CHP002 | 32,000,000 | David Ceperley | Quantum Monte Carlo Calculation of the Energetics, Thermodynamics and Structure of Water and Ice |
| CHM036 | 30,000,000 | Robert Harrison | CNP - Chemical Nanoscience at the PetaScale |
| FUS026 | 30,000,000 | Zhihong Lin | Gyrokinetic Particle Simulation of Transport Barrier Dynamics in Fusion Plasmas |
| NPH009 | 30,000,000 | David J. Dean | Ab Initio Structure of Carbon-14 |
| CLI030 | 28,500,000 | Kate Evans | Tests of Decadal Predictive Skill Using the Community Climate System Model |
| BIP008 | 25,500,000 | Jeremy Smith | Cellulosic Ethanol: A Simulation Model of Lignocellulosic Biomass Deconstruction |
| NFU003 | 25,000,000 | Dinesh Kaushik | Scalable Simuation of Neutron Transport in Fast Reactor Cores |
| CLI031 | 20,000,000 | Max J. Suarez | Explorations of High Impact Weather Events in an Ultra-High Resolution Configuration of the NASA GEOS Cubed Sphere Global Climate Model |
| CLI032 | 20,000,000 | Venkatramani Balaji | Petascale CHiMES - Coupled High-Resolution Modeling of the Earth System |
| FUS025 | 20,000,000 | Weixing Wang | Global Gyrokinetic Turbulence Simulations of National Spherical Torus Experiment (NSTX) |
| MAT015 | 15,000,000 | Mark Jarrell | Petascale Simulation of Strongly-Correlated Electron Systems Using the Multi-Scale Many-Body Formalism |
| TUR005 | 10,000,000 | Pui-kuen Yeung | Complex Transport Phenomena in Turbulent Flows: Leadership Computing at Extreme Scalability on the Cray XT5 |
| GEO003 | 9,500,000 | Peter Lichtner | Modeling Reactive Flows in Porous Media |
| NFI001 | 5,000,000 | Thomas Evans | Denovo, A Scalable HPC Transport Code for Multi-Scale Nuclear Energy Applications |
| CHM037 | 4,000,000 | Christopher Mundy | The Free Energy of Transfer of Hydronium from Bulk to Interface: A Comprehensive First Principles Study |
| FUS024 | 4,000,000 | Jeff Candy | Steady-State Gyrokinetic Transport Code (SSGKT), an SAP to the FACETS project |
| NTI010 | 3,800,000 | Lin-Wang Wang | Charge Patching Method for Electronic Structures and Charge Transports of Organic and Organic-Inorganic Mixed Nanostructures |
| LGT004 | 2,500,000 | Robert Sugar | Lattice QCD |
| CPH002 | 2,000,000 | Dario Alfe | AQUA |
| GEO004 | 2,000,000 | Omar Ghattas | Understanding the Dynamics of the Earth: High Resolution Mantle Convection Simulation on Petascale Computers |
| NEL002 | 2,000,000 | Gerhard Klimeck | Towards Petascale Simulation of Nanoelectronic Devices |
| TUR009 | 2,000,000 | George Vahala | Lattice Algorithms for Quantum and Classical Turbulence |

The Early Science projects on the XT5 are listed in Table 3.2. Twenty-six projects actively utilized the XT5, consuming over 350 million CPU-hours over the course of 6 months. Comparatively, the utilization is two and a half times greater than the entire 2008 INCITE usage, and three-quarters of the INCITE allocation. Fig. 3.8 shows the utilization by each project broken down by job size. The high utilization by materials science researchers is due in large part to significant development of the application DCA++, which allowed scaling to the full size of the XT5. Not only was it used for an Early Science project, but CDA++ was also the winner of the 2008 ACM Gordon Bell Prize for outstanding achievement among high-performance computing applications.

In Fig. 3.8, projects are listed in order of utilization, with largest users listed at the top and utilization in millions of CPU-hours shown to the left of the project IDs. The largest usage was from material

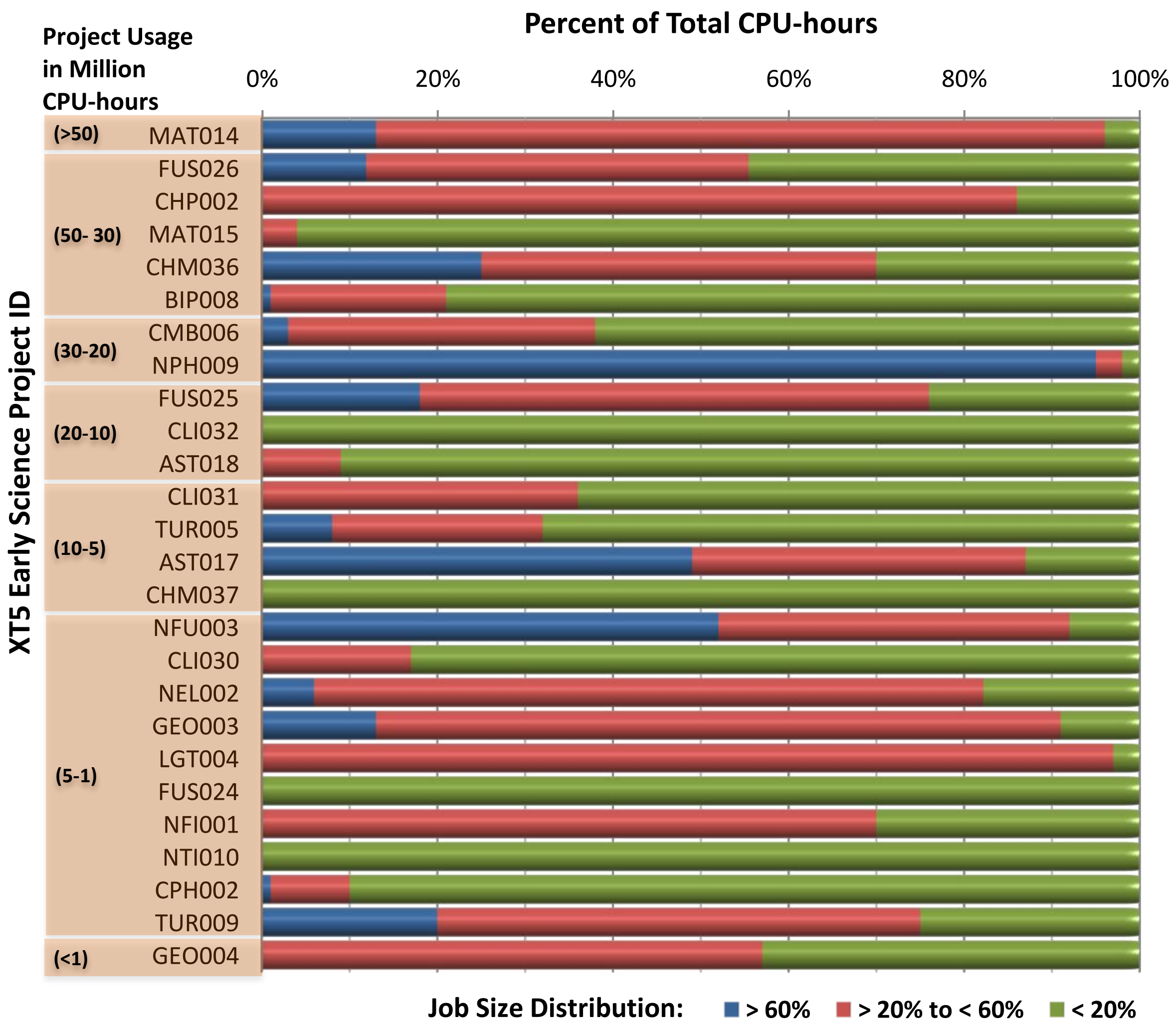


**Fig. 3.9. Utilization of Jaguar XT5 by job size for Early Science projects**

science (MAT014), with over 95% of their jobs utilizing greater than 20% of the XT5. Fourteen out of twenty-six projects exhibited clear leadership-class utilization with more than 50% of their usage at greater than 20% of the resource. Twelve projects were below standard capability utilization over all, with only four projects unable to scale adequately, exclusively at job sizes less than 20% of the XT5. It is interesting to note that three projects (AST017, NFU003, NPH009) in astrophysics, nuclear fusion, and nuclear physics used greater than 60% of the resource for roughly 50% or more of their utilization. Of those three, NPH009, the nuclear physics project, using the MFDn application (more details on applications are given in Appendix D) to perform ab initio calculations of carbon-14, utilized nearly all of their allocation for job sizes exceeding ~90K cores.

## 3.2.4 Scaling

To provide further understanding of the data from Fig. 3.8, Table 3.3 shows the largest core count used by various applications from INCITE and Early Science projects. Of the 28 applications listed, 27 use more than 20% of the XT5 and 19 codes use more than 60% for a single calculation.

**Table 3.3. Maximum scaling to date of INCITE codes**

| Science Category | Research Area | Code | Cores Used |
|---|---|---|---|
| **Biology** | Biology | GROMACS | 149,472 |
| **Chemistry** | Chemistry | NWChem | 96,000 |
| | | MADNESS | 140,000 |
| | | QMCPACK | 131,072 |
| **Computer Science** | General purpose lattice-boltzmann | Ludwig | 131,072 |
| **Earth Science** | Seismology | SPECFEM3D | 149,784 |
| | | FD3D | 32,000 |
| | Weather | WRF | 150,000 |
| | Climate | POP | 18,000 |
| | | CCSM | 80,000 |
| | | GEOS-5 Cubed Sphere | 62,208 |
| | Geosciences | PFLOTRAN | 132,000 |
| **Engineering** | Combustion | S3D | 210,000 |
| | | Senga | 131,072 |
| | Fluid Dynamics | SBLI | 130,000 |
| **Fusion** | Fusion | GTC | 153,600 |
| **Materials** | Materials | DCA++ | 213,120 |
| | | LSMS | 223,232 |
| | Nanoscience | LS3DF | 147,456 |
| | Condensed Matter | CASINO | 40,000 |
| **Nuclear Energy** | Nuclear Energy | Denovo | 57,600 |
| | | UNIC | 136,576 |
| **Physics** | Lattice Gauge Theory | BQCD | 65,536 |
| | Molecular Dynamics | CP2K | 32,768 |
| | Astrophysics | Chimera | 131,072 |
| | Nuclear Physics | MFDn | 220,000 |

## *3.3 Summary*

The usage statistics presented in this chapter show that for most science domains, users exhibit a need for leadership-class computing resources. This is demonstrated with the large core counts for a single calculation and the large number of overall CPU-hours needed to accomplish the science objectives. The need is growing from year to year as scientists are called on to address more and more challenging problems impacting the nation. The statistics also show that most application teams are reasonably adept at using current systems and capable of scaling their applications within a reasonable time to utilize changing resources.

# 4. SCIENCE APPLICATION PERFORMANCE

Accurate development of requirements for next-generation leadership computational science is based on an understanding of the behaviors of science applications used to perform this science. This chapter presents performance analysis of key OLCF applications, to show how performance of these applications drives demands for system hardware and software.

Assessment of application performance must be based on performance measures. At the highest level, the goals of next-generation high performance computing are to solve new science problems that could not be solved before (capability) and also to solve current problems faster and more cost-effectively (capacity). These computing tasks impose performance requirements on next-generation systems that can be quantified by cost factors such as:

- Time-to-solution for targeted application runs at specific core counts;
- Core-hours required for targeted application runs;
- Power consumption;
- Hardware constraints, such as main memory and offline memory capacities.

Evaluating science demands on future HPC systems requires estimating application performance for realistic future system workloads. Accurate prediction of future application performance enables better-informed procurement and planning decisions.

The OLCF application performance modeling process follows these steps:

1. Select heavily-used applications, based on utilization data from current workloads and estimation of future workloads.
2. Understand from domain scientists and developers the likely changes in application models, algorithms, science problems and use cases going forward.
3. Perform application benchmarks for representative problems on existing hardware to isolate the application stress points related to different hardware features.
4. Extrapolate this performance data to future architectures, based on anticipated hardware changes as well as changing science demands, to estimate future application performance.

The OLCF workload emphasizes leadership computing based on a comparatively small set of applications for targeted science areas. This limited number of applications makes it possible to narrow the focus and thus perform more in-depth application studies that give more meaningful estimates of workload performance for future hardware.

## 4.1 Application Performance Modeling Procedure

The OLCF performance modeling process is used to estimate performance of an application on a future system based on performance of the application on current systems. The first step of the application modeling process is to determine the proper application code version and to define a representative set of run specifications and core counts with the desired science and model features that represent typical workloads.

The application code is then compiled and run for the selected settings, using one or more profiling tools to extract runtime information. These tools include the standard FPMPI profiler [Gropp and Buschelman 2004], the customized FPMPI profiling tool developed by Cray, Inc., and the CrayPat profiler. These profilers provide data such as breakdown of runtime into wallclock time, scheduler time, communication time, synchronization time and input/output (I/O) time; detailed data on message sizes and counts; and, PAPI [Mucci et al. 2009] counter data. To obtain these statistics, multiple runs of the same case may be performed to check for run-to-run timing variability for each test case.

Though profiling tools are able to provide substantial run information, some application performance data must be obtained indirectly. To extract key information that is not readily available from profiling tools, selected additional runs are performed with slightly changed run settings, to measure the impact of certain hardware features. For example, a single test case at a fixed core count and identical problem settings is run multiple times using different numbers of CPU cores per processor socket, to isolate the impact of memory traffic on runtime due to different numbers of cores competing for the same memory. Similarly, for hybrid MPI/OpenMP or MPI/PThread codes, the same test case can be run at the same CPU socket count and MPI task count but different thread counts, to evaluate the threading efficiency and degree of thread parallelism.

The resulting performance data is then submitted to a performance model that estimates the execution time of the code on the proposed system based on hardware differences between the actual and targeted architectures. When the differences between current and targeted hardware is small, e.g., increasing the clock speed for the same processor type, the estimation process has relatively low risk. For larger hardware differences, more detailed analyses are required, e.g., inspection of the application source code or use of a simulator to understand the performance of code on the targeted hardware platforms.

To manage the process of performance modeling, the OLCF has developed the PMC performance modeling tool. This is a Python script library that automates operations such as downloading and building each application code, submitting the required jobs to the platform's job scheduler, monitoring execution, harvesting and analyzing run data, and generating performance reports based on this data.

## *4.2 Performance Indicators*

Application performance depends on a variety of hardware characteristics, such as processor clock speed, peak flop rate, cache structure, cache latencies, main memory bandwidth and latency, cores per compute node, communication bandwidth and latency and I/O bandwidth.

To condense the effects of these hardware factors on application performance into a usable form, several aggregate performance metrics are used, including:

- Fraction of runtime spent in each of CPU, memory, communication, and I/O.
- Flop rate percent of peak. This measure is important for some scientific applications, though obviously incomplete as a measure of CPU performance since some true CPU work is not flop-related (e.g., address computations).
- Floating point computational intensity, i.e., floating point operations per memory reference. This partly captures memory locality and memory reuse by the CPU in a relatively hardware-independent way, though it does not in itself measure reuse of data from the various caches.
- Communication bytes per second and messages per second, and their interrelationship. These measures are not hardware-independent, since they depend on the platform, core count, and problem size; nonetheless, they do help characterize in general terms the application communication requirements for representative runs on current hardware.

## *4.3 Components of Application Performance*

Using the performance modeling procedure described above, it is possible for representative runs of selected applications to distinguish the relative runtime costs associated with different hardware components, including CPU, memory subsystem, communication subsystem and I/O subsystem. Fig. 4.1 shows for selected OLCF applications the relative fraction of total runtime consumed by CPU, memory, communication and I/O at representative core counts (for application details, see Appendix D). These data were generated using the OLCF Jaguar platform, a Cray XT5 system using quadcore processors. For this set of test runs, application I/O was kept to a minimum, in order to reduce run-to-run variability due to the multi-user environment and thus obtain more reliable estimates. Fig. 4.1 shows usage of machine resources not only by application and science area but also by core count, indicating how performance scales.

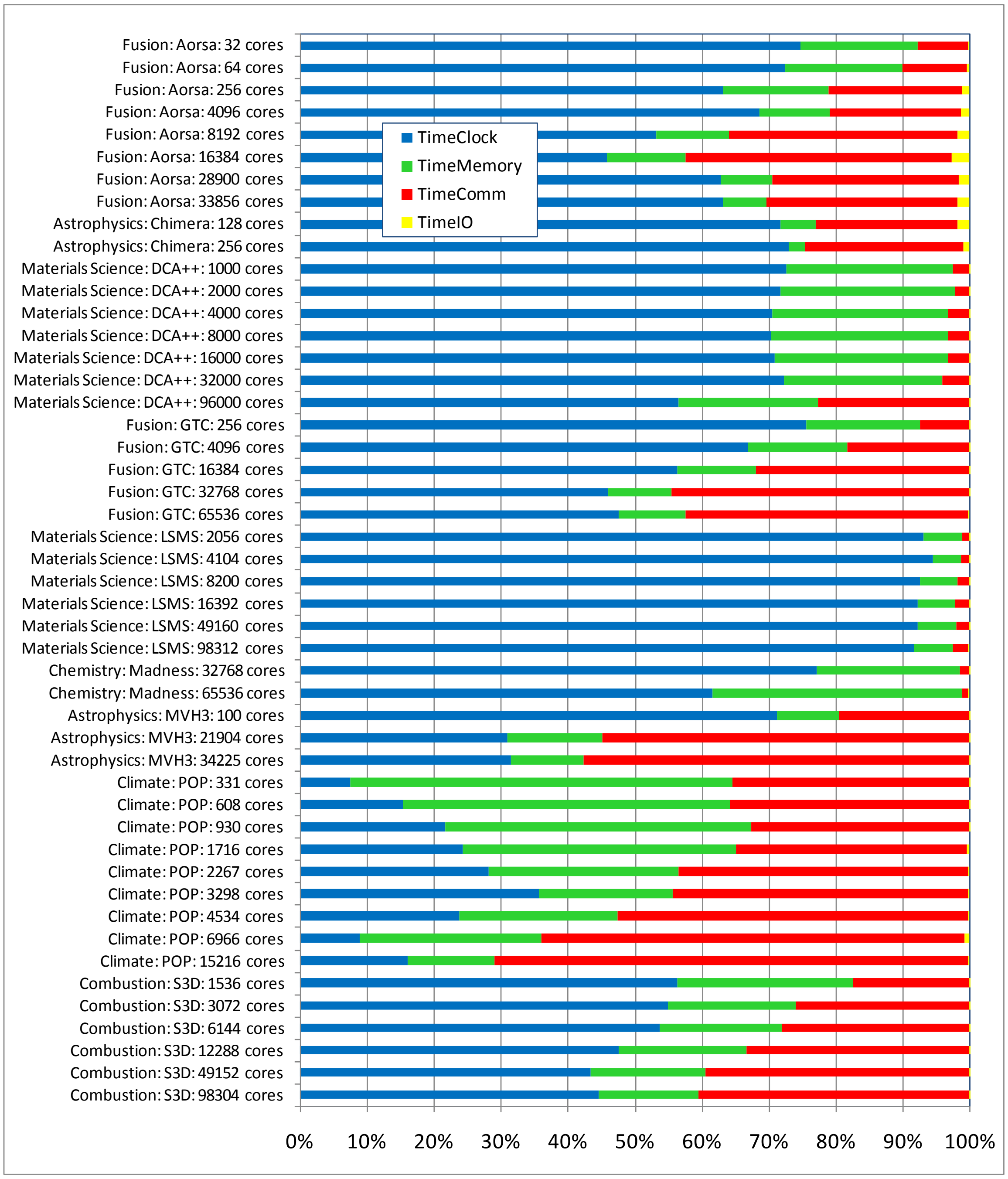


**Fig. 4.1. Application runtime fraction by hardware subsystem**

It is clear from these results that:

- Science applications such as LSMS, DCA++, AORSA and Chimera, which rely heavily on dense linear algebra, have a great deal of compute-bound work that can potentially be sped up by using more or stronger processor cores.
- Grid-based codes such as POP, S3D and GTC have significant communication requirements. Particularly, the POP code that is run in a strong scaling regime has communication requirements that increase as a fraction of runtime with increasing core counts.
- Overall, it can be seen that a great deal of processor-intensive work exists across the applications, which can be positively impacted by greater CPU capabilities of future leadership-class HPC systems.

The application runtime footprint for different hardware subsystems can also be analyzed in terms of the performance impact resulting from increasing the performance of a single hardware subsystem in isolation.

Fig. 4.2 gives Kiviat diagrams for selected application codes showing the relative impact on time to solution from improving processor, memory or communication performance by a factor of two. It is evident that the performance of codes like LSMS, DCA++, AORSA and Chimera is improved by more or stronger processor cores, and that codes like POP are improved by a better communication subsystem.

## *4.4 Computational Intensity and Percent of Peak*

As mentioned earlier, two measures of how effectively applications make use of computer hardware are computational intensity and percent of peak flop rate. Fig. 4.3 and 4.4 show computational intensity and percent of peak floating point operations for selected application runs on the OLCF quadcore Jaguar Cray XT5 platform.

Computational intensity is a measure of how much computational work is done by the CPU on a data value once the value is fetched from memory to a register. Codes such as DCA++, LSMS and AORSA with high computational intensity due to reliance on dense linear algebra make good use of the CPU, since performing many compute operations on a data value amortizes the cost of the memory access. On the other hand, codes like S3D, as is typical of grid-based codes, stress the memory subsystem more heavily and thus attain lower computational intensities.

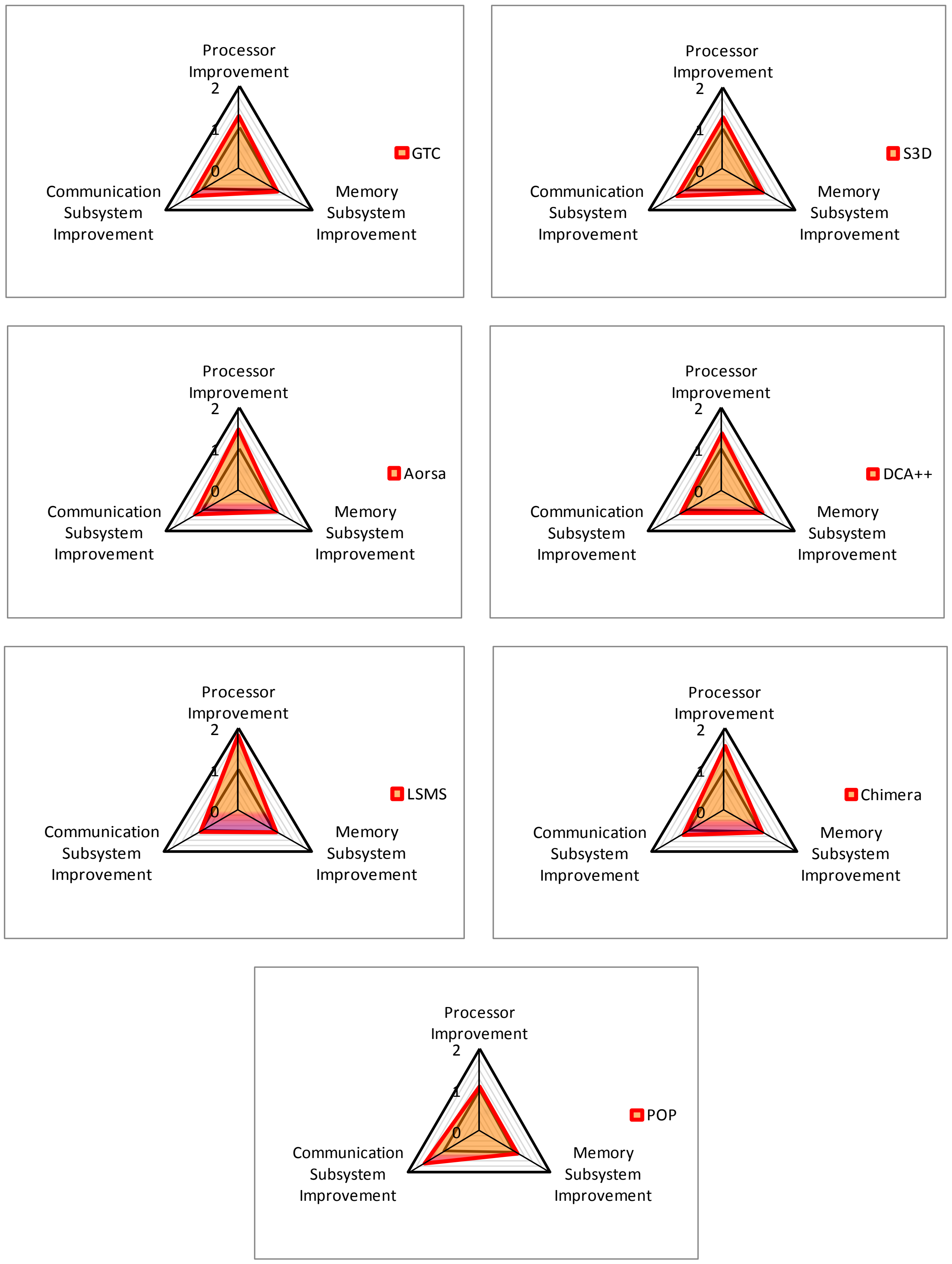


**Fig. 4.2. Impact of 2X hardware subsystem improvement on application performance**

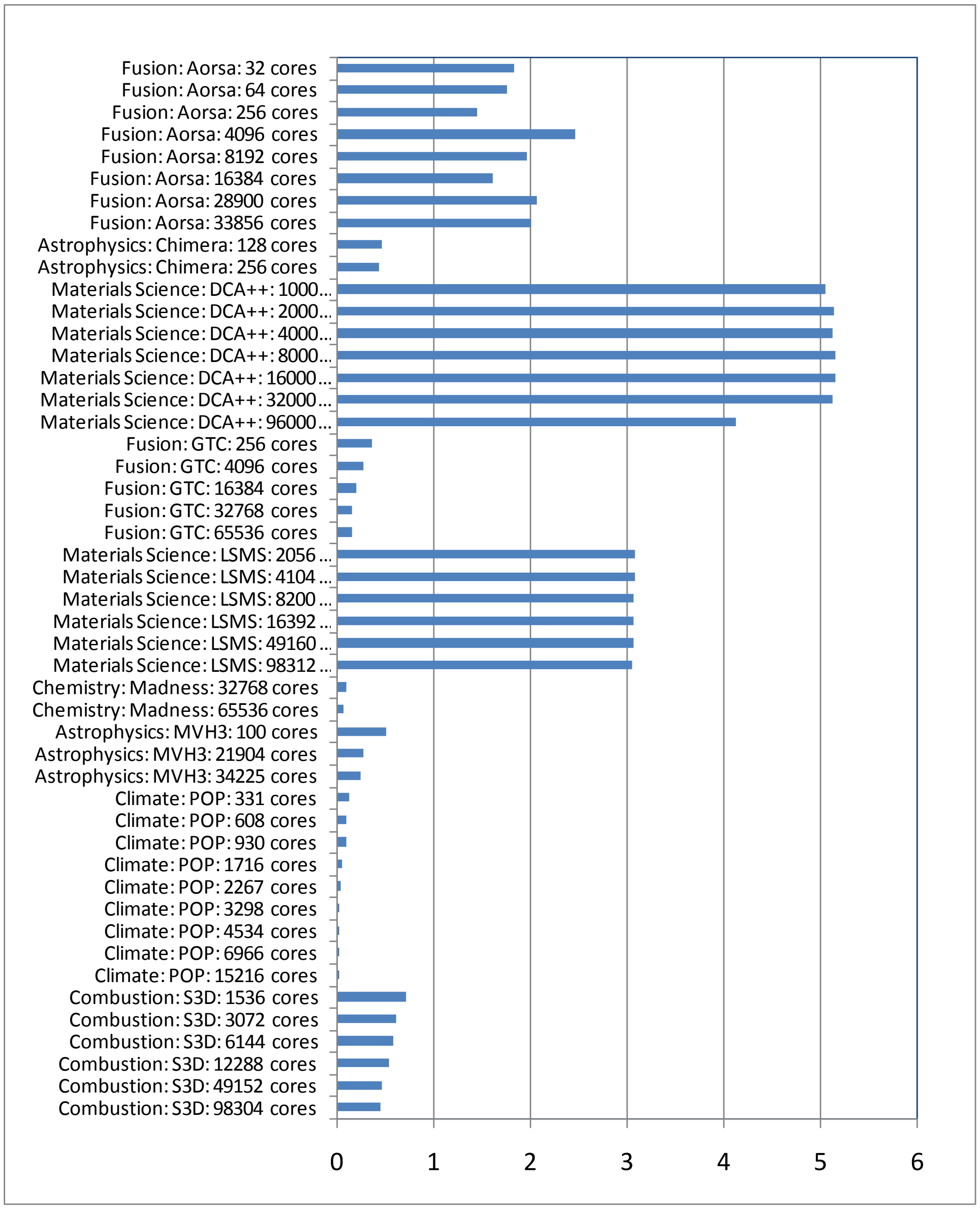


**Fig. 4.3. Application floating point operations per memory reference**

The range of application percent of peak has a broad range, with high values (e.g., 80%) for applications that heavily use dense linear algebra. The causes of lower fractions of machine peak speed are evident from Fig. 4.1: application runtime is spent not only in the CPU but also in memory access and communications. Unfortunately, many applications in practice are unable to exploit more than a small fraction of the available power of the CPU. This behavior is widely recognized to be a common

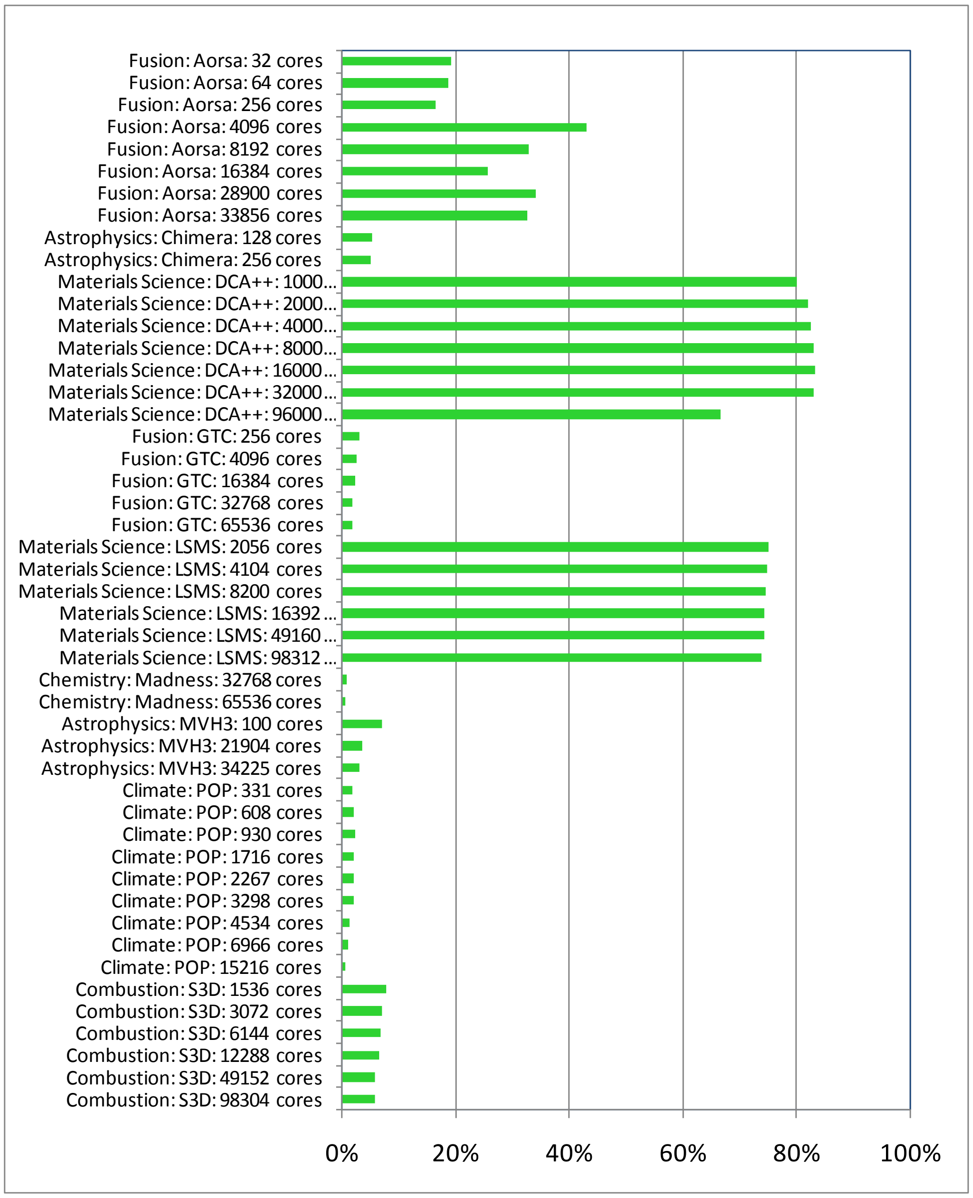


**Fig. 4.4. Application flop rate percent of peak**

occurrence for science applications in a wide range of areas, due to the "memory wall." For a number of years, the annual growth rate in memory speed has lagged the growth rate in processor power by about 30% per year [Graham et al. 2005, p. 108]. Thus, even the best written codes that have inherently memory-bound physics models will necessarily attain only small fractions of machine peak flop rate. This phenomenon adds urgency to the need to reformulate algorithms and science models along the lines of increasing locality and thus mitigating the effects of the memory wall.

Application performance can be characterized in terms of stress on CPU peak flop rate or alternatively on memory bandwidth. Fig. 4.5 shows the “roofline model” [Williams et al. 2009] comparing the computational intensity to percent of peak flop rate for selected application runs. In this graph, applications tend to be either compute bound (limited by the horizontal line representing maximum attainable flop rate) or memory access bound (limited by one of the diagonal lines representing peak bandwidth of the respective memory subsystem in the memory hierarchy). It can be seen that LSMS, DCA++ and AORSA are more compute-bound and that GTC and S3D are more memory-bound. POP is executed in a strong scaling regime, with its subproblems localized to L2 cache; however, it is communication-intensive and thus attains a low fraction of machine peak.

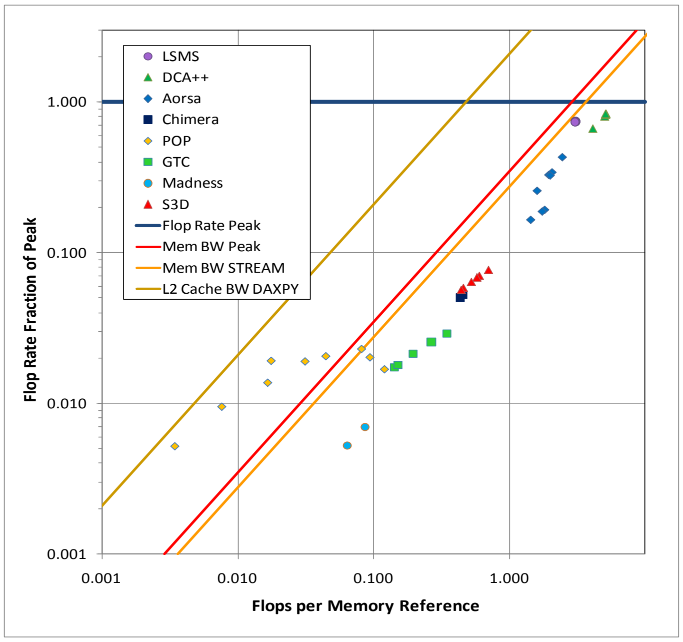


**Fig. 4.5. Application computational intensity vs. fraction of peak**

## *4.5 Communication*

Interprocessor communication network performance is a limiting factor for the performance of many algorithms. Application interprocessor communication requirements can be characterized by two measures: the number of messages sent per unit time, which stresses communication network latency, and the total amount of data communicated per unit time, which stresses network bandwidth.

Fig. 4.6 compares the communication bytes per second and messages per second for selected application runs on the OLCF quadcore Jaguar Cray XT5 system. Applications that send many messages per second (and thus tend to stress the latency limits of the communication subsystem) appear on the right side of the graph. Applications that send large amounts of data per second (and thus tend to stress the bandwidth limits) appear in the top part of the graph. From this it is shown that

- Codes like POP have high message rates, contributing to their being message latency limited.
- Codes like S3D and GTC which send 3-D halo data to neighbor processors have relatively high bandwidth requirements.

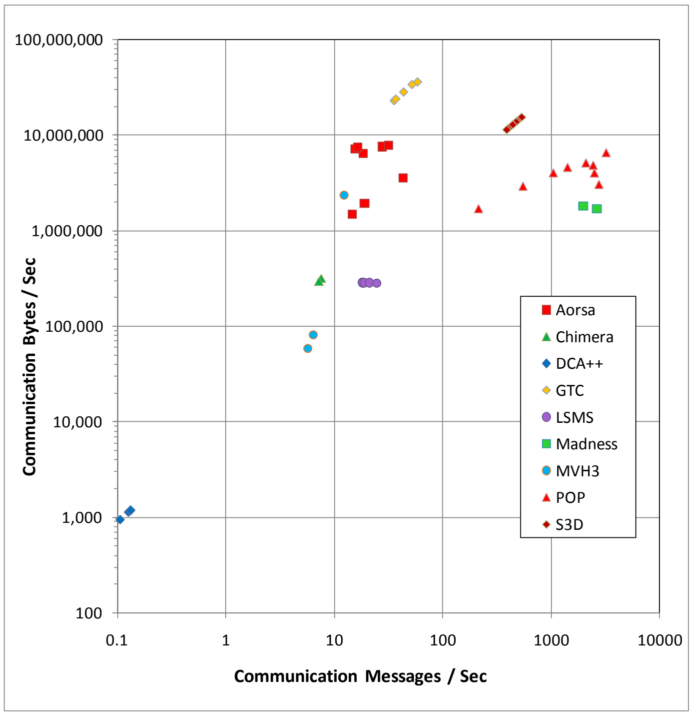


**Fig. 4.6. Application communication measures**

## 4.6 I/O

While parallel I/O performance requirements of applications taken in isolation are readily achieved by modern parallel file systems, aggregate I/O performance requirements on large-scale simulation environments remain difficult to meet. For such large-scale environments the parallel file system is a shared resource. Parallel I/O environments on petascale class platforms are made up of tens of thousands of disk drives, hundreds of I/O servers, and increasingly complex system area networks to deliver the aggregate bandwidth and I/O operations per second (IOPs) required by the diverse workloads these platforms support. To provide a productive simulation environment, parallel I/O systems must be designed to support increasingly diverse workloads such as "metadata intensive" workloads in which tens of thousands of files are created per second, "IOPs intensive" workloads in which hundreds of thousands of application processes are generating small (e.g. 4KB) I/O requests concurrently, and "bandwidth intensive" workloads in which thousands of application processes are generating large (e.g. 1MB) I/O requests concurrently. Under these mixed workloads, I/O performance is often limited by metadata operations and IOPs rather than by aggregate theoretical bandwidth, as larger I/O requests are interleaved with much smaller I/O requests creating contention. Under IOPs-intensive workloads as seen on many large-scale simulation environments, I/O bandwidth drops to a small fraction of the theoretical bandwidth achievable under ideal workloads.

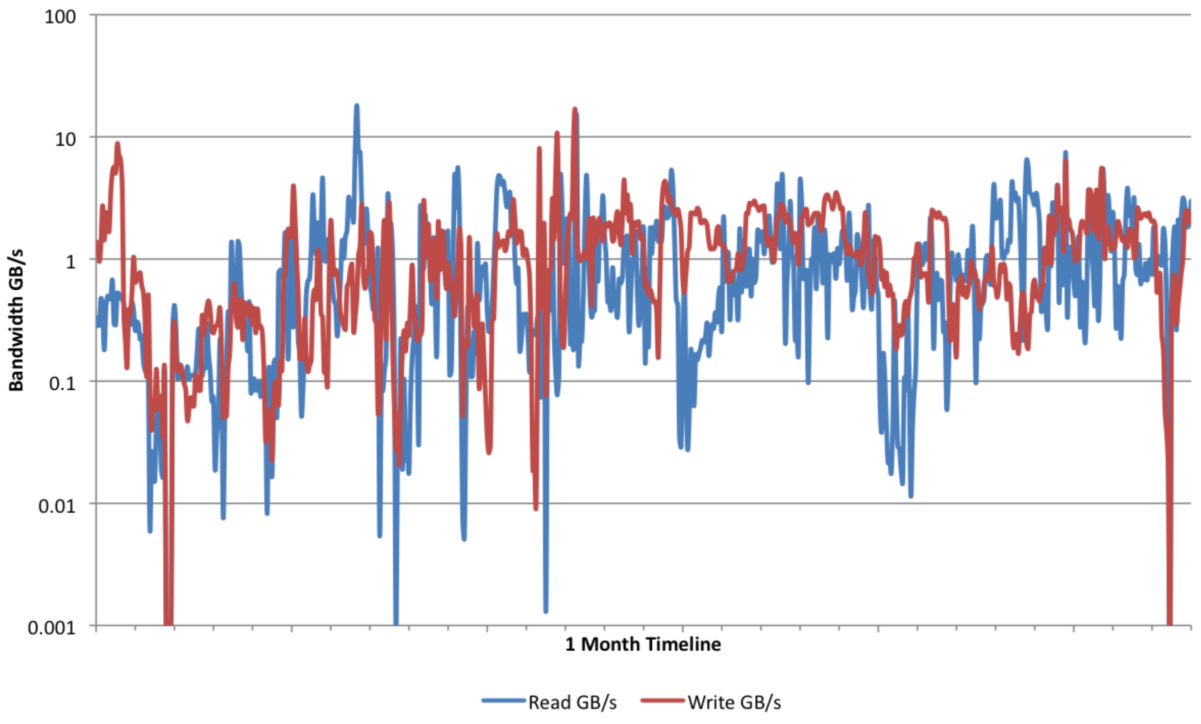


**Fig. 4.7. Aggregate I/O bandwidth (mixed workload)**

Due to the shared nature of the parallel I/O environment, we have focused our performance study on aggregate system load on the underlying storage system. Data presented in this study is obtained from a live large-scale production system over a one-month period of time. Fig. 4.7 illustrates the aggregate load (in terms of I/O bandwidth, gigabytes per second) on the underlying I/O subsystem generated by a diverse application mix. While average system load is relatively low, the ability to deliver high parallel I/O performance under heavy workloads is critical for providing a productive simulation environment and avoiding sustained periods of system performance degradation, demonstrated by the dramatic peaks in delivered system performance.

As detailed above, mixed workloads result in extremely high IOPs in the parallel I/O environment. Traditional storage technologies such as magnetic disk drives are susceptible to extreme performance degradation in terms of bandwidth under small I/O, high IOPs workloads. Fig. 4.8 illustrates the aggregate load (in terms of IOPs, thousands of operations per second) on the underlying I/O subsystem from a diverse application mix over the same one-month period analyzed in Fig. 4.7.

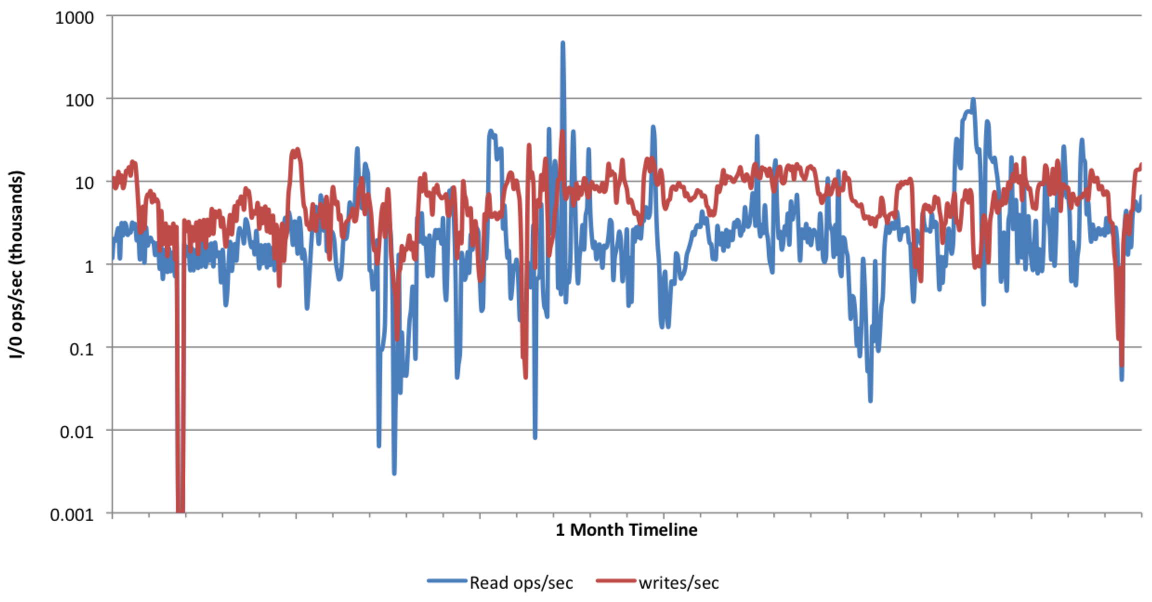


**Fig. 4.8. Aggregate IOPs (mixed workload)**

Fig. 4.7 and 4.8 together tell us that the ability to deliver high IOPs is as important as the ability to deliver high parallel I/O performance towards achieving a productive parallel file system for large-scale simulation environments.

## 4.7 Power Consumption

The power demands for computer hardware have become a major cost factor for executing applications on HPC systems. Modern processors are able to adapt their power utilization based on the calculations they perform at any given time. The growing concerns over power consumption suggest the importance of including power consumption as a metric for optimizing algorithm design. Science applications can be designed not just to minimize floating point operations or memory references but also power-related metrics such as “science results per watt” [Kogge 2008, p. 228]. Similarly, an understanding of power consumption behaviors of applications can inform future computer hardware designs.

Fig. 4.9 shows power consumption for dedicated application runs on the quadcore Jaguar Cray XT5 system at selected core counts. Unsurprisingly, DCA++ and AORSA, which heavily use dense linear algebra kernels, have high computational intensity and thus high power consumption per core. On the other hand, less compute-intensive codes such as S3D have high power requirements as well. Studies of this type can be used to shed light on costs measured in terms of power as well as compute time associated with particular algorithm designs.

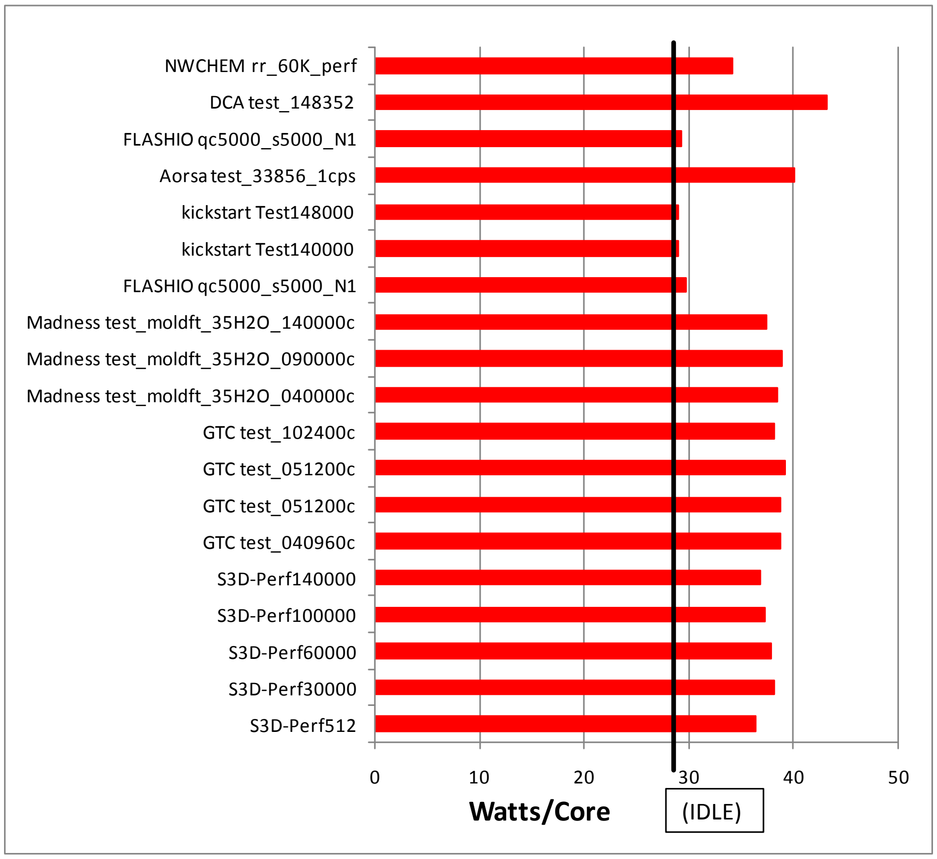


**Fig. 4.9. Application power consumption**

## *4.8 Implications for Future HPC Systems*

The performance characteristics of OLCF applications as described above indicate how these applications may be expected to perform on future hardware and motivate the requirements for future leadership HPC systems.

Anticipated changes in HPC system hardware include: (1) increasing demands for memory locality, and (2) the need for applications to express increased thread parallelism, even to tens of billions of threads

[Kogge 2008, p. 198]. These hardware trends and the above performance results for OLCF applications motivate the following observations.

- Science applications relying heavily on compute-intensive kernels, such as several of those described above, are well positioned to adapt to future hardware changes, since they already have a high degree of locality. In particular, accelerator technology (see Section 5.6) will benefit these applications due to their high flop rates.
- Other applications must increase the amount of data locality. More broadly, a locality-centric orientation should be promoted across the parallel algorithm research and development community. Though the memory bottleneck has grown for over a decade, fundamental algorithmic research and deployment have been slow to respond. Algorithmic design must focus on doing as much meaningful computational work with local data as possible, in algorithm areas beyond dense linear algebra. In this algorithm design regime, some design decisions might lead to reordering of computations at the algorithm level or even the performing of redundant computations, if warranted by the tradeoff of computation for memory reference. Deployment of such optimizations should be strengthened by raising the community awareness level regarding the importance of these optimizations as well as evaluation of how such code modifications could be implemented in realistic code development processes and maintained in code bases without impeding progress in more science-related code development activities.
- Algorithmic innovations such as lower precision and mixed precision arithmetic can improve both memory reference costs and floating point operation costs. This approach is already in use in one of the codes considered here (DCA++).
- Certain application codes already have significant potential for more thread parallelism, e.g., DCA++, LSMS, S3D and GTC. This should be exploited.
- Communication latency-bound codes such as POP should use algorithms that implement better latency hiding techniques, e.g., iterative solvers using polynomial preconditioning or s-step Krylov methods [Joubert and Carey 1992, Demmel et al. 2007]. Such measures cannot entirely eliminate the need for local or global communications, so vendor hardware must continue to address latency issues.
- Similarly, some applications should be rethought from first principles in terms of whether the limits to parallelism are intrinsically imposed by the physics of the simulation or whether the limit is an artificial result of a negotiable algorithm choice. For example, an explicit time-

dependent grid-based PDE code might be able to take several time steps between communications without altering the numerical answer, by doing larger halo communications and performing some redundant computations on each processor with this data.

- Communication bandwidth-bound codes might be adjusted so that communication better matches the underlying hardware. For example, recent experiments with S3D show that total performance can be improved by over 20% by MPI task remapping.

- Importantly, there is no single magic bullet hardware feature for improving application performance across the board. Different applications in different science areas have different hardware stress points, such as computation, memory, communication bandwidth and communication latency. Next-generation HPC systems cannot afford to neglect any of these hardware features. When possible, trends in the commodity hardware market can be leveraged to address these needs. When this is not possible, alternative hardware solutions must be developed. Notably, accelerator technology will address several of these hardware concerns, including node peak flops, memory bandwidth and memory latency (see Section 5.6). Other factors such as interconnect bandwidth and latency must be improved as well.

- The programming model for many of these applications is MPI-only, though this is changing in the direction of MPI plus threading. For example, Madness and GTC already implement some form of threading in addition to MPI. Further modifications of this type must take place, to better exploit coming hardware changes that are geared toward fine-grained parallelism. Appropriate code changes, whether at the kernel level or at the large-scale code restructuring level, need to be considered for increasing parallelism and breaking operations into smaller units of independent work.

- A better understanding of the relationship between algorithm and power consumption can better inform a cost model that includes all relevant factors including time to solution, consumed core-hours and power utilization as objectives for optimization.

- Better software tools could aid in the performance optimization process. This might include improved performance profiling tools that have more lightweight functionality but are more robust across hardware vendors, programming languages, compiler vendors and parallelism models. Also, usability issues regarding these tools should be addressed—performance analysis tools could be made more accessible to non-specialist application developers by giving more attention to usability engineering issues [Nielsen 1993].

In short, we make the following summary recommendations:

1. **Hardware:** OLCF applications stress multiple hardware features; no single hardware feature can be improved to the neglect of others.
2. **Parallelism:** Substantial efforts must be undertaken to restructure applications and codes to prepare for multibillion-thread parallelism.
3. **Latency:** A much more latency-centric approach to algorithm and software design is required.
4. **Optimization:** Better adaptation of codes to communication networks and memory hierarchies can substantially improve performance.
5. **Power:** More efforts to optimize the power consumption of algorithms and applications should be undertaken.
6. **Tools:** More robust and usable performance analysis tools can assist application developers in improving code performance.

# 5. SCIENCE APPLICATION REQUIREMENTS

This chapter discusses the elements required to perform next-generation leadership science. The requirements for leadership science are of two types: resource requirements and process requirements. Resource requirements are generally easier to quantify; process requirements are more qualitative, being more dependent on institutional experience and empirical research. In either case, it is the goal of the OLCF requirements process to make science application requirements increasingly quantifiable and objective, thus improving the accuracy and reliability of the planning process.

Computational resource requirements for science applications include computer hardware, system software stack, scientific libraries, compilers, parallel programming interfaces, computational algorithms and models. Processes include application development management, application workflow management, software quality and assurance, verification and validation, and data management. In what follows each of these items is addressed in turn.

## *5.1 Science Model Requirements*

The DOE Office of Science has determined that significant increases in computational capabilities are required to address serious economic, environmental and national security challenges. As mentioned earlier, these vital needs for increased computational capabilities will enable advances in basic science in numerous technical areas with a broad range of societal impacts.

Advancing science in these key areas requires development of next-generation physical models to satisfy the accuracy and fidelity needs for targeted simulations. The impact of these simulation fidelity needs on requirements for computational science is twofold. First, more complex physical models must themselves be developed to account for more aspects of the physical phenomena being modeled. Second, for the physical models being used, increases in resolution for key system variables, such as numbers of spatial zones, time steps or chemical species, are needed to improve simulation accuracy, which in turn places higher demands on computational hardware and software.

Application models represent the functional requirements that drive the need for certain numerical algorithms and software implementations. The choice of model is in part motivated by the science objectives, but it is also constrained by the computer hardware characteristics attainable in the relevant time frame. The choice and specification of system attributes (e.g., peak speed or node memory capacity) tend to constrain the functional attributes able to be employed in a given physical model on that system.

**Table 5.1. Model requirements for next-generation science**

| Science Area | Application | Future Model Requirements |
|---|---|---|
| Astrophysics | CHIMERA | Increase in number of spatial zones. Increase in number of nuclear species from 17 to 150-300, increasing the compute time for nuclear burn by 500x or more, thus increasing full simulation cost by 20x or more. |
| Bioenergy / Biology | LAMMPS | Simulation of microbial/biomass interface and dynamics of enzyme action on biomass. Higher atom count, longer time scales. Additional force fields, more accurate potentials. Scaling by >10x to increase accuracy. |
| Chemistry | MADNESS | Couple new components to model. Use more accurate component models. |
| Climate | CAM-HOMME | Incorporate full land surface model, next-generation microphysics, radiative transfer, aerosol indirect effect, full biogeochemistry. Include better representations to parametrize subgrid scale physics, advanced time stepping routines to improve speed and accuracy, new discretization schemes and grids, more coupled prognostic variables. |
| Combustion | S3D | 50–100% increase in Re (from 10K to 15–20K) with corresponding up to 8x increase in grid resolution and increase in the number of time steps. Alternatively increase number of chemical species from 22 to 60-80 with corresponding 3-4x increase in computations. |
| Fusion | AORSA | Incorporate finite width particle orbits in plasma response calculation. |
| Fusion | GTC, GTS | Add coupling of core and edge physics, late time evolution, neoclassical turbulence physics. |
| Materials Science | DCA++ | Extend to multi-band model which scales the cost by order $n^3$ in the number of bands for at least 8x cost increase. Possibly increase from 24 to 32 atoms, which scales the computation by order $n^3$. |
| Materials Science | WL-LSMS | Extend the model to non-spherical potentials and more accurate density functional which could increase cost by 10x. Increase atom count which scales the cost linearly. Increase number of walkers by 10x to calculate additional observables to increase accuracy of result. |
| Nuclear Energy | Denovo | Model full BWR cores; Gen-IV designs; experimental facilities. Increase the number of energy groups from 44 to 999 (23x), which will in turn demand further resolution in space and angle. Modeling of uncertainty. |

For example, attributes such as the following all depend in part upon the hardware system for which implementation of the models is targeted:

- model state variables (how many now, how many planned in the future);
- model characteristics (partial differential equations [PDE] or ordinary differential equations [ODE], deterministic or stochastic, formulation of equations);
- primary model data types (double precision floating point, integer);
- presence of multiple, simultaneous phenomena, and the required degree of coupling;

- presence of multiple scales in multiscale models;
- domain of dependence (local with specific patterns, global); and
- data dependency (degree of parallelizability).

As part of the requirements solicitation process, the OLCF consulted with over 50 leading scientists in numerous science domains, to better understand the goals for next-generation science and the models that would be required to reach these goals.

Table 5.1 summarizes key findings from this process. It can be noted that for several science areas, the science demands require a significant quantum leap forward in science model capabilities. Furthermore, several science areas are able to quantify the factor of growth in problem resolution required to meet their science goals. Projects able to quantify this factor are shown to require growth of an order of magnitude or more for their computational needs. In particular, current hardware capabilities on the order of 2 petaflops must be extended to 20 petaflops or more to accommodate near-term science requirements.

To reach the science goals mandated by DOE, action must be taken to provide solutions for this increase in demand for high-end computing capabilities.

## 5.2 Computational Algorithm Requirements

Science priorities lead to science models, and models are implemented in the form of algorithms. Algorithm selection is based on various criteria, such as appropriateness, accuracy, verification, convergence, performance, parallelism and scalability.

Models and associated algorithms are not selected in isolation but must be evaluated in the context of the extant computer hardware environment. Algorithms that perform well on one type of computer hardware may become obsolete on newer hardware, so selections must be made carefully and may change over time.

As mentioned earlier, moving forward to exascale will put heavier demands on algorithms in at least two areas: the need for increasing amounts of data locality in order to perform computations efficiently, and the need to obtain much higher factors of fine-grained parallelism as high-end systems support increasing numbers of compute threads. As a consequence, parallel algorithms must adapt to this environment, and new algorithms and implementations must be developed to extract the computational capabilities of the new hardware.

**Table 5.2. Algorithm requirements for next-generation science**

| Science Area | Application | Algorithm Requirements |
|---|---|---|
| Astrophysics | CHIMERA | CURRENT: Finite-volume (PPM) hydrodynamics, multi-group flux-limited diffusion neutrino transport, fully-implicit solution of nuclear kinetics networks. Structured grid, adaptive mesh redistribution, sparse linear algebra, passive particles, GMRES solver with custom preconditioner.<br>FUTURE: Quasi-equilibrium (QSE) methods, full 1D Boltzmann transport, possibly AMR. |
| Bioenergy / Biology | LAMMPS | CURRENT: Molecular dynamics,multiple integrators, velocity Verlet , Charm22 force field and others, bonded and non-bonded terms, charge terms.<br>FUTURE: New models, scalable algorithms for long-range forces. |
| Chemistry | MADNESS | CURRENT: Adaptive multiresolution methods, low separation rank representations of functions/operators, validation against experimental results.<br>FUTURE: Increasing accuracy of calculations, solving larger problem sizes with more accurate physics models. |
| Chemistry | NWCHEM | CURRENT: Density functional theory (DFT), Hartree-Fock with LCAO , second order Moller-Plesset theory (MP2), coupled cluster CCSD(T), other quantum chemistry correlated methods.<br>FUTURE: Enhancements for IR and Raman spectra of large molecules, additional correlated methods. |
| Climate | CAM-HOMME | CURRENT: Cubed sphere, spectral discretization, with finite volume and spectral element (continuous and discontinuous Galerkin), explicit time stepping.<br>FUTURE: Fully implicit time stepping, iterative linear solver. |
| Combustion | S3D | CURRENT: Higher order accurate finite difference, fully explicit RK time integration with error estimates and time step control.<br>FUTURE: Possibly adaptive chemistry, load balancing. |
| Fusion | AORSA | CURRENT: Double precision complex dense solve, quasi-linear operator, CQL3D coupling for calculation of particle distribution functions.<br>FUTURE: Mixed precision solver, replacement of CQL3D for more accuracy . |
| Fusion | GTC, GTS | CURRENT: Particle-in-cell simulation solving gyrokinetic equation in Lagrangian coordinates using a delta-F method, general geometry using spline fit of MHD equilibrium and experimental profiles data, global field-aligned mesh using magnetic coordinates, Monte Carlo method for collisions.<br>FUTURE: Full-F. |
| Materials Science | DCA++ | CURRENT: Dynamic cluster quantum Monte Carlo algorithm with Hirsch-Fye auxiliary field solver with delayed updates.<br>FUTURE: New delayed update algorithm, continuous time quantum MC solver. |
| Materials Science | WL-LSMS | CURRENT: Calculation of scattering path matrix for complex energies by iterating the block inverse formula.<br>FUTURE: Larger problem sizes. |
| Nuclear Energy | Denovo | CURRENT: Inverse power iteration for eigenpair calculation, accelerated with coarse mesh finite diffusion, inner solves with 2-grid preconditioned Gauss-Seidel, within-group solves using DSA-preconditioned GMRES.<br>FUTURE: Shifted inverse power iteration, Krylov inner solves over all energies, energy groups in inner loop. |

As with science models, the performance of algorithms can change in two ways as application codes undergo development and new computer hardware is used. First, algorithms themselves can change, motivated by new models or performance optimizations. Second, algorithms can be executed under different specifications, e.g., larger problem sizes or changing accuracy criteria. Both of these factors must be taken into account.

**Table 5.3. Employment of algorithm motifs by science areas**

| 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 | | | |
| | OReTran | X | | 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 | | |
| | Denovo | X | | | X | X | | |
| Nuclear physics | NUCCOR | | | | X | | | |
| QCD | MILC | X | | | | | | X |

The OLCF requirements elicitation process was used to poll domain scientists regarding algorithm requirements going forward for the respective application codes. Results for key applications are presented in Table 5.2. These results should be read in tandem with the data in Table 5.1 regarding predicted changes in problem size parameters. These results show substantial algorithm development requirements to support new science models as well as anticipated computer hardware changes.

The impact of these algorithm changes on application performance can be understood in terms of a selected set of algorithm motifs [Colella 2004,Asanovic et al. 2006]. These algorithm motifs, such as structured and unstructured grids, FFT, dense and sparse linear algebra, particles and Monte Carlo methods, occur repeatedly across a wide range of science applications. Table 5.3 shows the usage of these algorithm motifs in OLCF applications. Analyzing application codes in terms of these application motifs helps identify common requirements across applications as well as common stress points for computer hardware.

**Table 5.4. Anticipated change in use of algorithm motifs for next-generation science**
(↑ = increasing usage, ↓ = decreasing usage)

| Science Area | Application | Algorithm Motif Change |
|---|---|---|
| Astrophysics | CHIMERA | Structured grids, dense linear algebra (↑), sparse linear algebra (↑), particles, unstructured grids (↑) |
| Bioenergy/ Biology | LAMMPS | FFT (↓), particles (↑), structured grids, dense linear algebra(↑), sparse linear algebra(↑) |
| Chemistry | MADNESS | Unstructured grids, dense linear algebra (↑) |
| Chemistry | NWCHEM | FFT, dense linear algebra |
| Climate | CAM-HOMME | Structured grids, particles, sparse linear algebra (↑) |
| Combustion | S3D | Structured grids (↑) |
| Fusion | AORSA | Structured grids, FFT, dense linear algebra (↑) |
| Fusion | GTC, GTS | Structured grids, sparse linear algebra, particles (↑), Monte Carlo |
| Materials Science | DCA++ | Dense linear algebra (↑), Monte Carlo (↑) |
| Materials Science | WL-LSMS | Structured grids, dense linear algebra |
| Nuclear Energy | Denovo | Structured grids (↑), sparse linear algebra (↑), dense linear algebra (↑) |

Table 5.4 gives for selected applications the algorithm motif used and, when appropriate, the anticipated growth of usage of these algorithm motifs in the science applications going forward. This increased usage is reckoned in terms of new algorithm development, platform node-hour usage or fraction of total runtime for typical runs.

It should be noted from Table 5.4 the anticipated increased usage for many algorithm motifs represented. This mandates the need for greater support for optimized software that efficiently performs the requisite mathematical operations in support of these motifs.

This information motivates the following recommendations:

- As mentioned earlier, algorithm research is essential to the success of applications on next generation hardware. Algorithm research should be driven by the need for more locality-centric algorithm variants with more available fine-grained parallelism to address upcoming hardware changes. The increasing reliance on key algorithmic motifs by OLCF applications suggests efforts focused specifically on the motif components that are heavily used across multiple applications. Encapsulating software for these motifs within libraries when appropriate allows code optimization efforts to be leveraged across multiple projects.
- Support should be given to parallel software library development in these and related vital motif areas. These software libraries must be optimized for the cases required by the codes of domain scientists. Unfortunately, in many cases available software libraries are not optimized for problem cases of importance to specific applications (e.g., a dense BLAS kernel may not be optimized for a matrix shape that is uncommon in general situations but is important to a specific science application area). In the course of library development, often it is impossible to foresee the range of input settings or slight algorithm variations that might be associated with key science application runs. This problem must be addressed by an appropriate software strategy such as, (1) when possible provide compiler support to generate high-quality code for the required operations, perhaps with the help of compiler hinting techniques via directives; (2) support library developers to implement the required functionalities directly in the library when needed; (3) make the library source code open source and thus more easily customizable by others to the required problems; and (4) use adaptive methods or autotuners when appropriate to optimize for a broader range of cases.

## *5.3 Parallelization and Compiler Software Requirements*

### 5.3.1 Near-Term Requirements

Application development teams routinely select programming language, compiler and parallel programming model based on a variety of factors such as model and algorithm structure, compiler performance, supported platforms, team expertise and needs to reuse legacy code. Table 5.5 gives for selected OLCF applications the choice of programming language, compilers, communication libraries and lines of maintained code.

**Table 5.5. Languages and parallel programming models for key applications**

| Science Area | Application | Programming Language(s) | Compilers Supported | Comm Libraries | LOC |
|---|---|---|---|---|---|
| Astrophysics | Chimera | F77, F90 | PGI, Cray, IBM | MPI | 250K |
| Bioenergy / Biology | LAMMPS | C++ | GNU, PGI, IBM, Intel | MPI | 140K |
| Chemistry | MADNESS | C++ | GNU | MPI, PThreads | 1M |
| Climate | CAM-HOMME | F90 | PGI, Lahey, IBM | MPI | 500K |
| Combustion | S3D | F77, F90 | PGI | MPI | 10K |
| Fusion | AORSA | F90 | PGI | MPI | 20K |
| Fusion | GTC | F90, C, C++ | PGI, Cray | MPI, OpenMP | 8K |
| Materials Science | DCA++ | C++ | GNU | MPI | 26K |
| Materials Science | gWL-LSMS | F77, F90, C, C++ | PGI, GNU | MPI | 70K |
| Nuclear Energy | Denovo | C++, Fortran, Python | GNU, PGI, Cray, Intel | MPI | 46K |

It is clear that both FORTRAN and C++ are used heavily for application development, with no clear winner. Furthermore, projects typically use an MPI-only parallel programming model, with some use of hybrid approaches.

The requirements elicitation process also revealed trends that are expected to change this snapshot of application development approach. In particular, numerous teams are moving toward hybrid programming models, adding some type of threading such as OpenMP, PThreads or Intel TBB to their current MPI framework. This is motivated by performance or memory usage improvements that are expected to result.

### 5.3.2 Next-Generation Requirements

Seismic changes currently underway in parallel computing hardware, such as heterogeneous processors, multibillion-threaded platforms and ever-increasing relative latency times, are motivating the drive for new programming models to exploit advances in hardware capability. In particular, approaches such as OpenCL and new directives-based compiler technologies are being considered.

Development of new programming interfaces for these hardware innovations are in the early stages. A period of user experimentation and experience-gathering will be required before a consolidation around best practices is possible. Since these are early stages, specifications cannot yet be given in detail regarding software requirements to match the new hardware. Still, a number of required criteria for new programming interfaces for next-generation hardware are manifest:

- The interface must allow good performance for the average scientific programmer and allow access to a high percentage of attainable peak performance for the expert performance-oriented programmer.
- The compilers and libraries must be robust and stable.
- A standardization process and widespread availability across current and future platforms are necessary to protect programmer investment in application code development.
- The interfaces must have relative ease of use with good performance for general parallel scientific programmers.
- Multiple programming interfaces (e.g., threads, message passing, one-sided messaging) may be required to meet the needs of a diverse set of science approaches.
- Interfaces must support hardware heterogeneity (e.g., multicore processors and accelerators in the same hardware environment).
- Interoperability is required, to support mixed or hybrid programming models in the same code, e.g., MPI+OpenMP or directives+OpenCL. Mixed models are often required for supporting tradeoffs in performance tuning and incremental code parallelization.
- The programmer must be able to control process locality, data locality and data motion, e.g., whether memory is allocated on processor memory or accelerator memory.
- When possible, languages and interfaces should support the reuse of legacy code. Reuse of code can often reduce software development costs by a factor of three to four and can also substantially reduce defect rates [Kandt 2006, p. 203; Grady 1992, p. 14].
- Asynchronicity of memory and compute operations should be supported when appropriate. Similarly, work sharing between heterogeneous compute units should be permitted.
- Parallel programming interfaces should allow for incremental code parallelization, though it is recognized that in many cases substantial code restructuring or rewriting may be necessary to obtain good performance.

- Compilers should support as many standard language features as possible (e.g. a C++ compiler with parallelization directives should not omit important standard language features such as templates).
- Programming interfaces should be adaptable to new forms of heterogeneity that might arise in the future.

## *5.4 Scientific Library Requirements*

Science applications typically make use of one or more scientific or mathematical software libraries. Use of scientific software libraries is advantageous on several fronts: it saves developer time by offloading relevant algorithm implementation and code optimization work from the science programmer; it in principle provides better performance on the targeted hardware; and it focuses developer technical expertise onto areas where it is best used.

Table 5.6 shows scientific library usage across a wide range of OLCF applications. It is clear that applications make heavy use of a number of third-party mathematical libraries. More tellingly, Table 5.7 gives estimates of the fraction of runtime spent in mathematical libraries for several key heavily-used OLCF applications. It can be inferred from these tables that the key applications spend on average over 30% of their time in scientific libraries. This heavy usage provides a good opportunity to leverage optimization efforts for a few key libraries, to provide broad increases in performance across many science domains.

For libraries to be effective for OLCF users, they must satisfy several criteria. They must be well-supported by the vendor or third-party developer. They must be easily interfaced to the user code with minimal invasiveness to the application. They must also be heavily optimized for the targeted hardware and for the application use cases.

### 5.4.1 Expanding the Range of Coverage for Scientific Libraries

As mentioned above, key OLCF applications spend on average about 1/3 of their time in scientific library calls. However, this leaves another 2/3 of runtime untouched. Developing libraries to address this fraction of runtime would be of substantial value, from the standpoint of both development time and application performance. The relative value of this will be even greater going forward, as hardware and science models become increasingly complex.

To address this need, we recommend an effort to consider a broader range of commonly-used operations or "application kernels" as potential candidates for implementation in library form. This would extend the coverage of libraries from the limited scope of lower-level functions to higher-level mathematical or even physics-level operations.

**Table 5.6. Application library usage**

| Science domain | Code | I/O libraries | Math libraries |
|---|---|---|---|
| Accelerator design | T3P | NetCDF | MUMPS, ParMETIS, Zoltan |
| Astrophysics | CHIMERA | HDF5 (pNetCDF) | LAPACK |
| | VULCAN/2D | HDF5 | PETSc |
| Biology | LAMMPS | ------------ | FFTW |
| Chemistry | MADNESS | ------------ | BLAS |
| | NWChem | ------------ | BLAS, ScaLAPACK, FFTPACK |
| | OReTran | ------------ | LAPACK |
| Climate | CAM | NetCDF | (SciLib) |
| | POP/CICE | NetCDF | ------------ |
| | MITgcm | NetCDF | ------------ |
| Combustion | S3D | ------------ | ------------ |
| Fusion | AORSA | NetCDF | HPL, ScaLAPACK, FFTPACK, PGPLOT |
| | GTC | MPI-IO, HDF5, NetCDF, XML | PetSC |
| | GYRO | MPI-IO, NetCDF | BLAS, LAPACK, UMFPACK, MUMPS, FFTW (SciLib, ESSL) |
| Geophysics | PFLOTRAN | ------------ | BLAS, PetSC |
| Materials science | LSMS | HDF5, XML | BLAS, LAPACK |
| | QBOX | XML | LAPACK, ScaLAPACK, FFTW |
| | QMC | ------------ | BLAS, LAPACK, SPRNG |
| Nanoscience | CASINO | ------------ | BLAS |
| | VASP | ------------ | BLAS, ScaLAPACK |
| Nuclear energy | NEWTRNX | HDF5 | LAPACK, PARPACK |
| Nuclear physics | CCSD | MPI-IO | BLAS |
| QCD | MILC, Chroma | ------------ | ------------ |

Various efforts in the past have attempted to create higher-level libraries or frameworks for general use, often with mixed or poor results. Numerous libraries and component frameworks have been developed to good effect in the context of individual projects or smaller user communities but have often failed to become widely usable in general. Broader acceptance has failed for a variety of reasons, such as functionality too restrictive, required use cases not covered, required user data formats not supported, performance not well-optimized, software design too invasive to user code, package methods too tightly coupled or inflexible, interfaces not standardized, software not adequately supported, package not available on targeted platforms, or usage learning curve too steep. In short, often the use of the library or framework provides little or no payoff for the effort of attempting to use it.

**Table 5.7. Estimated mathematical library runtime fraction**

| Science Area | Application | Libraries | Percent Runtime |
|---|---|---|---|
| Astrophysics | Chimera | LAPACK | 78% |
| Bioenergy / Biology | LAMMPS | FFTW | 5-75% |
| Chemistry | MADNESS | LAPACK | 50-75% |
| Climate | CAM-HOMME | Trilinos solver (future) | (large) |
| Combustion | S3D | (N/A) | |
| Fusion | AORSA | HPL, ScaLAPACK | >35% |
| Fusion | GTC | (N/A) | |
| Materials Science | DCA++ | LAPACK | 70% |
| Materials Science | gWL-LSMS | LAPACK | 70-75% |
| Nuclear Energy | Denovo | Trilinos | small |

At the same time, a well-chosen, carefully implemented set of application-centric software microkernels could have high impact on numerous science applications. "Development of key mini-applications that represent the performance-determining computations of key application areas is important." [Geist et al. 2007] In fact, the "algorithm motif" concept [Colella 2004] itself carries with it the suggestion that certain well defined operations taken from the motif areas might be good candidates for implementation in general-purpose libraries, to the degree that algorithm structures and use cases are mature and stable. Examples might be particle pushing operations for particle-in-cell codes, sparse matrix-vector product operations, finite element assembly for FEM codes, and stencil computations for structured grids.

Code reuse in the form of libraries is notoriously lacking in some of these areas. For example, bandwidth-optimized methods for stencil computations, though having high performance impact in numerous situations, have been repeatedly rediscovered and reinvented over the last 15 years, with little to no success in preserving institutional memory of their existence. Since problems like this have been solved repeatedly, an effort should be made to make the solutions more widely known and generally available.

We recommend that next generation scientific application middleware libraries have the following characteristics:

- **Fine-grained:** Put simply, each user-callable library component or function call should perform a single responsibility. In other words, the functions should be small, recombinable building blocks. This design feature converts the need to cover a prohibitively large scope of potential use cases into the ability to access a combinatorial space of arrangements in which the pieces can be joined at will by the user. This design philosophy is analogous to the preference of RISC over CISC instruction set architectures in the microprocessor design world.
- **Decoupled:** Functions must be capable of being used independently, and they must have well-defined, simple interactions with the outside world. This is a requirement for the easy combination of component functions.
- **Use case driven:** Kernels must be chosen so that they can be effectively used in the actual application use cases of interest to the community. For example, the BLAS-1 kernels were carefully chosen to cover a significant scope of vector computation situations in common use [Lawson et al. 1979].
- **Flexible:** It has been said that a truly great tool has uses beyond the scope of its original design intent [Raymond 1999]. Library components should be adaptable to the new situations scientists frequently devise. Examples might include different data types or data structures or slightly different boundary conditions. A mechanism for incorporating this flexibility might be by the use of generic programming techniques.
- **Tuned:** The library must be highly optimized for the targeted hardware (e.g., using autotuning techniques). In some cases it may be desirable to include the autotuning software with the library code itself, to allow developers to tune the library to new hardware or new application use cases.
- **Benchmarked:** Library users must be able to predict how the library will perform under various settings or function combinations, in order to make informed design decisions regarding model and algorithm selection. Library documentation should be accompanied by performance statistics to make it possible to understand library performance for different cases. In addition, it might be required that a library benchmarking tool be shipped with the library, to aid with performance prediction.

- **Standardized:** Interfaces increase in value when they are agreed upon by multiple parties. Development and acceptance of standards will ease developer use and encourage more widespread support across multiple platforms.

## 5.5 System Software Requirements

The system software stack provides the software infrastructure and services required for science applications to be executed on HPC platforms. Table 5.8 illustrates the types of components in the software stack and typical required capabilities.

**Table 5.8. Software stack requirements**

| System software component | 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 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 5.9 shows specific software used to satisfy these requirements.

These tables indicate current software requirements. We anticipate future software requirements to change in the following ways:

- Better OS support for fault tolerance will be required, as mean time to interrupt (MTTI_ figures are pressured downward.

- New parallel programming interfaces such as CUDA and OpenCL will require software support in the form of compilers, debuggers, and emulators where appropriate.
- Source code analysis tools will be useful to help identify code sections amenable to fine-grained parallelization.
- Debuggers, profilers and libraries will need to support new forms of heterogeneous compute nodes.
- Compilers will need to support OpenMP and/or other directives to access relevant node hardware efficiently.

**Table 5.9. System software to satisfy software stack requirements**

| Requirement | OLCF software stack |
|---|---|
| Resource manager/scheduler | Torque, Moab, ALPS |
| Scripting tools | bash, Perl, Python, Tcl/Tk |
| Build tools | make, configure, autoconf, m4 |
| 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, GNU, Cray, Pathscale, Intel |
| Vendor math libraries | LibSci, 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 |
| Accelerator support | CUDA, OpenCL |
| Low-level communication layers | Portals, ARMCI, Global Arrays |
| Shared memory layers | OpenMP, PThreads |
| 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 |

## *5.6 Hardware Requirements*

Science applications are ultimately dependent on the underlying hardware for performance. A given OLCF system has many hardware attributes that uniquely characterize it relative to other systems. The following twelve attributes have been found to be particularly useful and important to consider from an application perspective:

1. Peak flops per node,
2. Mean time to interrupt,
3. Wide area network bandwidth,
4. Node memory capacity,
5. Local storage capacity,
6. Archival storage capacity,
7. Memory latency,
8. Interconnect latency,
9. Disk latency,
10. Interconnect bandwidth,
11. Memory bandwidth, and
12. Disk bandwidth.

The performance of computer hardware associated with these system attributes affects different algorithm classes in different ways, depending on the algorithm structure and computational requirements. Table 5.10 shows the impact of hardware system attributes on different commonly-used OLCF algorithm classes and applications.

Likewise, science application codes in science domain areas have differing requirement levels for different hardware attributes. Table 5.11 gives a three-tier ranking of the importance of each hardware attribute to each science area. For each science area, the demand placed on each hardware attribute is classified as high, medium or low.

**Table 5.10. Impact of system attributes on algorithms and applications**

| System Attribute | Algorithms that Require High Performance for this Attribute | Relevant Application Behaviors |
|---|---|---|
| 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. |
| 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. |

**Table 5.11. Importance of hardware attributes to science domains**
(Priority: Red = high, pink = medium, grey = low.)

| 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 | 🟧 | 🟧 | 🟧 | 🟧 | ⬜ | 🟧 | 🟧 | ⬜ |

It is important to understand how future science needs and concomitant science model changes will influence the demands on each of these hardware characteristics for future HPC platforms. Table 5.12 shows for each science area the anticipated change in demand for hardware attributes.

The highlighted rows of Table 5.12 indicate the hardware characteristics that are expected to grow in importance for OLCF applications on future systems. To prepare for the demands that science applications are expected to put on hardware, actions such as the following will be required:

- **Interconnect bandwidth:** For future systems, the power cost to transport data off-chip will be an increasing challenge [Kogge 2008, p. 212]. For the longer term, new hardware technologies will be needed to address this problem. Bandwidth-reducing and communication-hiding algorithm research can help mitigate this problem.

- **Node peak flops:** Since performance gains from clock speed and instruction level parallelism have reached an impasse, other approaches such as multicore processors and accelerators will be required to increase flop rates.

**Table 5.12. Future changes in demand for hardware characteristics**

| System Attribute | Climate | Astrophysics | Fusion | Chemistry | Combustion | Accelerator physics | Biology | Materials science | Total |
|---|---|---|---|---|---|---|---|---|---|
| Node peak flops | + | + |  | + | + | – | + | + | **+5** |
| MTTI |  | + |  |  |  | + |  | + | **+3** |
| WAN network bandwidth | – | – | + | + |  | + | – | – | **-1** |
| Node memory capacity | + | + | + |  | - | + |  |  | **+3** |
| Local storage capacity |  | + | – |  | – |  |  |  | **-1** |
| Archival storage capacity |  |  | – |  |  | – |  | – | **-3** |
| Memory latency | + | – |  | – | + |  | + | + | **+2** |
| Interconnect latency | + | – |  | – | – | + | + | + | **+1** |
| Disk latency | – |  | – |  | – | – | – | – | **-6** |
| Interconnect bandwidth | + | + | + | + | + |  | + |  | **+6** |
| Memory bandwidth | + |  | + |  | + |  | + | + | **+5** |
| Disk bandwidth |  |  | – | + | – | – | – |  | **-3** |

- **Memory bandwidth:** The "memory wall" has increased in severity for many years. On the algorithm side, the memory wall can be addressed in part by more locality-aware algorithms. On the hardware side, bandwidth-optimized accelerators in the form of streaming processors hold promise for gains (see below).

- **MTTI:** Hardware reliability and fault tolerance will continue to increase in importance. For the shorter term, hardware must be designed so that any single hardware failure will have only local impact, i.e., will not cause a failure of the whole system. This can be addressed for example by making use of adaptive hardware. For the longer term, it is likely that new techniques will be required to address this concern, at the level of hardware, system software and application programming models [Kogge 2008, p. 3]. Resiliency will be a major concern for attaining exascale, since standard disk-hased checkpoint/restart techniques will become less viable due to MTTI pressure from the sheer number of components used.

- **Node memory capacity:** Some applications will require increasing amounts of memory per core, in some cases to implement more complex and realistic physics models. New hardware will require "fat-memory" nodes that do not shrink the available memory per core.
- **Memory latency:** Improvements in hardware latency have not kept pace with improvements in bandwidth. Algorithms and application codes must be restructured to allow for more latency hiding. Furthermore, processor hardware must be able to hide more latency by allowing more in-flight memory references.
- **Interconnect latency:** Next-generation interconnect hardware must keep communication latencies under control and allow for high message injection rates. Furthermore, latency-reducing algorithms must be developed and implemented to reduce the impact of latency on application performance.

### 5.6.1 Accelerator Technology

As mentioned earlier, next-generation science goals will drive demand for an order of magnitude or more increase in computational capabilities. This growth is manifested most directly in the need for more aggregate floating point operations per second for next-generation systems, but science requirements also place demands on multiple hardware characteristics, as shown earlier.

For nearly twenty years, high performance computing has found it advantageous for reasons of cost to leverage the economies of scale provided by the commodity processor hardware market. However, conventional commodity processors are no longer able to produce performance gains through increases in clock speed or instruction level parallelism. Thus, the support of increasing numbers of processor threads has become a key avenue for increased performance of commodity hardware.

This factor has driven the growth of multicore processors. But more dramatically, it has given rise to a variety of many-core streaming graphics-accelerated chips suitable for general-purpose programming, such as the IBM Cell BE, AMD Fusion and Intel Larrabee processors, as well as general-purpose graphics processing units from companies such as NVIDIA.

Fig. 5.1 and 5.2 demonstrate the annual growth rate of peak processor performance and peak memory bandwidth for NVIDIA processors. The rate of performance growth demonstrated by these processors is dramatic, demanding the attention of future HPC system developers.

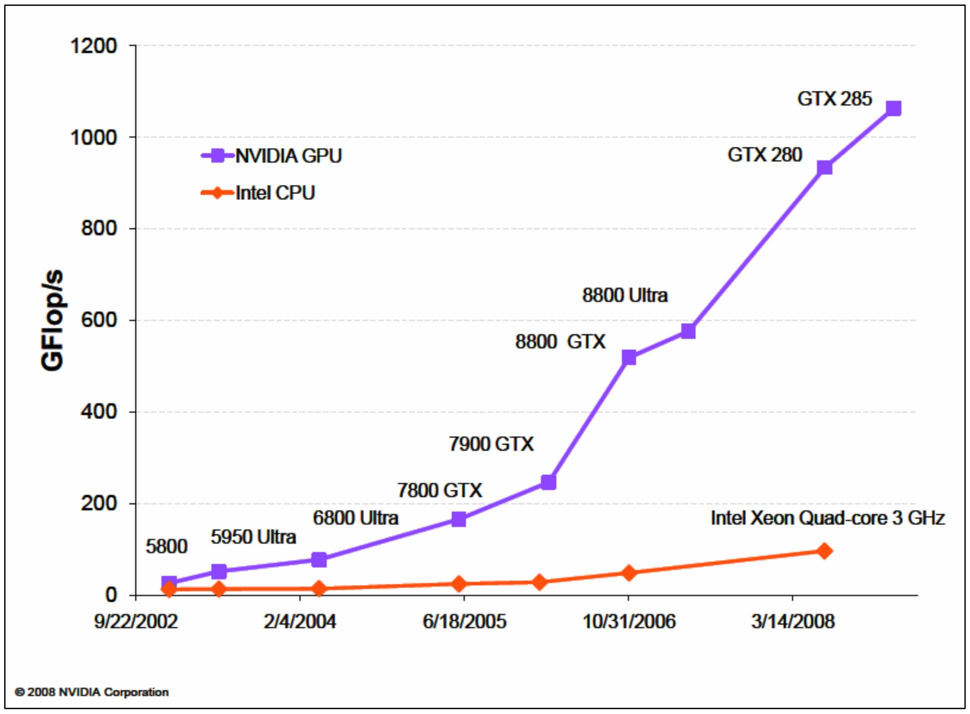


**Fig. 5.1. Growth rate of peak processor performance**

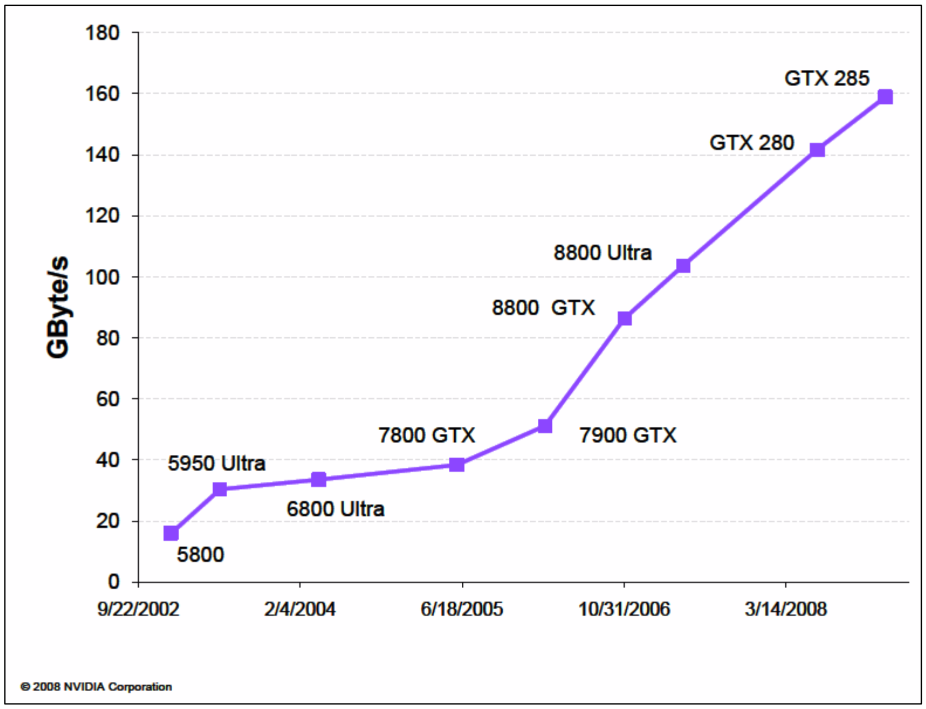


**Fig. 5.2. Growth rate of peak memory bandwidth**

Notably, accelerators of this type are able to address three of the seven hardware concerns for future application performance mentioned above:

- **Node peak flops**, due to more processors per die,
- **Memory bandwidth**, and
- **Memory latency**, due to the ability to hide memory latency with many outstanding in-flight memory references.

Next-generation systems must pay close attention to the potential performance gains offered by these accelerators. However, use of accelerator technologies for HPC is not without challenge. As mentioned earlier, parallel programming models are likely to experience a period of disruption as significant as the transition to message passing two decades ago, and programming methodologies will require a period of time to assimilate the changes. Software development teams must begin now to prepare for the requisite software disruptions this new hardware will impose. Some of the required preparations are discussed in the next section.

## *5.7 Application Development Process Requirements*

All progress in computational science is ultimately dependent on the processes of development of the science applications. In the future, this process is expected to become increasingly challenging, for various reasons:

- HPC computer hardware is becoming increasingly complex and heterogeneous.
- To access the performance potential of the hardware, it is becoming necessary in some cases to use multiple/hybrid programming models in the same application (e.g., MPI+OpenMP).
- Science models are becoming increasingly complex. Also, multiple codes are being composed together to generate new science.
- Core counts are becoming higher, making it more difficult to locate performance problems or execution errors.
- The increasing incidence of hardware faults or unusual hardware behaviors can make it more difficult to determine whether observed problems are due to the application, system software or hardware.
- The combined effect of these complexities makes it increasingly difficult for individual developers to maintain deep expertise in multiple requisite domains.

The survey results from science code teams using OLCF facilities indicated that currently the following are the primary bottlenecks to the application development process, in order of priority:

1. Debugging and testing;
2. Optimization and tuning;
3. Managing software development collaborations;
4. Learning new programming models.

In what follows we address these concerns.

### 5.7.1 Software Defect Reduction Tools

By far, the primary concern expressed by science application code teams concerns debugging and testing of application software. Based on the trends mentioned above, the challenges in this area can only be expected to increase. We recommend strong action be taken to address this issue, in two areas: improved tools and better software engineering practices.

Studies have shown that software defect detection and correction activities can consume as much as 50% of the labor effort to create software and 75% of total software life-cycle costs [Kandt 2006, p. 177]. Experience has shown that good software debugging and correctness checking software can substantially reduce this figure. For example, one study found that organizations using software tools for error detection delivered software with seven times fewer defects [Kandt 2006, p. 191].

Historically, debuggers for HPC systems have suffered from significant problems such as:

- inability to scale to high core counts;
- failure to present parallel code behavior information to the user in a meaningful way;
- lack of availability for specific targeted hardware and/or compilers;
- slow, unresponsive interactive behavior that greatly decreases programmer productivity; and
- lack of reliability across use cases and source code constructs.

A fundamental concern is that high-quality debuggers are not being delivered within the timeframe in which they are most needed. It is difficult to adapt debuggers and other tools to rapidly changing hardware and software. Top HPC hardware systems have short lifecycles that can be measured in terms of a few years. Major compiler releases occur at a rate of at least once per year, and minor releases occur even more frequently. New language or library features (e.g., Fortran 20XX), languages (e.g., UPC, Co-Array Fortran, X10, Fortress, Chapel) and programming models (e.g. OpenMP, OpenCL, MPI 3.0) continue to emerge. These factors slow the debugging tool software development process and thus make it difficult for debugging tools to be available and robust precisely during the time when they are most needed [Vetter 2007, p. 7].

Several alternatives are possible to address this problem. First, funding agencies can increase their support for tool development. Second, a healthier market, driven by increasing demand for debuggers scaling to high core counts, can provide a wider range of alternatives. Third, debugging tool developers can more directly treat the debugger code as a scalable application in its own right and improve their code designs accordingly. Developers of debuggers can also use software design methodologies that enable easier extension to new software and hardware environments. Finally, debugger development efforts can be refocused toward more lightweight, modular tools that can be more quickly and easily ported to new hardware and new compilers. In the absence of a full-featured interactive development environment which, due to its integration, would improve developer productivity substantially, the development of more easily ported, simpler debugging tools would provide more available functionality and timeliness for scientific application developers.

Additionally, a greater attempt should be made to leverage applicable commercial off-the-shelf (COTS) software. For example, memory checking tools such as Rational Purify [IBM 2009] applied to single core runs can be extremely effective for identifying bugs that are otherwise very difficult to locate. Unfortunately, it is often the case that "the HPC community is unaware of these tools … application developers either do not know about or do not have access to new tools" [Collette et al. 2004, pp. 1-3]. However, many tools have matured in the mainstream software development community; some of these can be leveraged for HPC programming efforts. In this way, in the area of software tools, as well as hardware, benefit can be derived from leveraging products from the larger mainstream market.

It should also be recognized that, to a large extent, "deep analysis is largely a user function, rather than a tool capability" [Vetter 2007, p. 7]. Debugging and testing capabilities must, to an increasing degree, be implemented directly within application codes. The criteria for what small pieces of data are required from the application to facilitate the user's debugging experience can be very complex, requiring either debugger support for very complex query expressions based on application code values or user-written debugging functions within the application. Developing code infrastructure to generate diagnostic information within the application can be of help.

### 5.7.2 Improved Software Engineering Practices

The benefits of good software engineering practices for scientific code development efforts are well-documented, and efforts to strengthen software quality within organizations can be of significant value. For example, through implementation of improved software quality practices, the NASA Software

Engineering Laboratory over a four year period decreased software defect rates by 75% and reduced costs by 55% within the organization [Kandt 2006, p. 44]. However, institutional efforts to improve software quality practices must be done carefully, since nearly two thirds of organizational efforts to improve software processes result in failure [Kandt 2006, p. 44].

Many factors that improve programmer productivity and reduce the occurrence of software bugs are fairly well-understood based on empirical research. This is true of both the software development community at large and the HPC scientific software development community in particular [HPCS DTWG 2007].

Many of these defect reduction practices are well-proven and should be implemented. For example, unit testing is well-known to increase programmer productivity by helping developers locate defects more quickly. Implementing unit testing in an application can significantly increase individual development time. However, since on average 20 percent of the modules of a software system account for 80 percent of the defects, and less than 5 percent of code can account for more than 45 percent of defects [Kandt 2006, p. 158], unit testing can be deployed strategically to particularly complex error-prone parts of the code, to limit the time required for deployment. Another effective practice is the use of software inspections, which can detect as many as 45-60 percent of defects [Kandt 2006, p. 178] and can reduce the cost of defect detection by 5-17 times compared to release testing only [Grady 1992, p. 161].

Though software quality practices are of value, historically some conclusions drawn from software engineering research regarding the broader software development community have not been entirely applicable to the niche industry of HPC scientific software development, for various reasons:

- The high importance of software performance in the computer time vs. developer time cost mix, compared with many PC applications for which software runtime performance is almost irrelevant.
- The difficulty of supporting some software quality principles in the face of immature and rapidly changing software and hardware environments.
- The more short-term research-oriented nature of some codes. It has been noted that for scientific software projects, "… each [software engineering] technique must be individually evaluated to match costs and benefits to project goals. It would be counterproductive, for example, to dogmatically apply the rigorous quality assurance processes necessary for mission-critical software—where one bug could crash a plane—to the development of scientific codes that thrive on risky innovation and experimentation." [Post and Votta 2005, p. 40]

- The need for flexibility of programming approach based on science domain and application type, for which a "one size fits all" approach to programming technique is not appropriate.

Nevertheless, many of the findings from software engineering research are of value to HPC application development efforts. However, such practices have been either slow in deployment or non-existent. For example, it has been remarked that in most computational science software development efforts, "few of even the simplest and best-known proven methods for organizing and managing code development teams are being employed" [Post and Votta 2005, p. 40].

To remedy this problem, actions such as the following can be taken as first steps to promote the diffusion of appropriate software engineering ideas within the community:

- Provide education in the form of training on HPC-aware software engineering practices. Many HPC practitioners have backgrounds primarily in the physical sciences and may have little previous exposure to software quality practices. Educational venues would increase programmer awareness of potentially useful methodologies for software development.
- Promote awareness and facilitate usage of mainstream software development tools when appropriate. For example, usage of interactive development environments are able to double programmer productivity [Kandt 2006, p. 170]; however, in some cases these tools do not run efficiently on front ends of HPC systems due to network latency times, thus reducing or eliminating any potential for productivity gains. Better tool support might remedy this problem.
- Foster a community of best practices that conserves experiences regarding the tradeoffs of code efficiency, code flexibility and programmer productivity.
- Provide institutional leadership and incentives to support improved software quality at the organizational level.
- Support a specific HPC-aware software engineering component within science application development teams.

In light of the inevitable heightening of the challenge in developing and debugging HPC application software, organizations must take seriously the need to improve institutional software quality efforts, in order to reduce risks regarding the success of next-generation systems and commensurately control costs.

### 5.7.3 Parallel Programming Interfaces

Factors such as the rise of heterogeneous computing, the quantum jump in the amount of parallelism required in applications, and the increasing need for locality are creating a disruption in the methods of programming HPC hardware. The potential performance gains for science applications are too great to ignore this paradigm shift. Various methodologies are being proposed for parallel programming APIs (MPI, OpenMP, OpenCL, CUDA, other directives-based methods, PGAS and HPCS languages, hybrids, etc.) [Kasim et al. 2008].

From the standpoint of science application development and deployment, the following are requirements for an effective new programming interface to next-generation systems:

- Parallel programming interfaces must be user-friendly to new developers but also permit access to high-performance hardware capability to the development code teams' HPC performance "power users". Optimized code within applications must at least be readable, and better yet maintainable, by science programmers who are not performance experts.
- Parallel programming interfaces must have stability, maturity, and planned long-term support, to reduce the risk level and attract users to write to the interface.
- There must be a small, bounded ratio between the performance of casually written source code using the programming interface and the performance of highly optimized source code. Otherwise, the programming interface must allow a high increase in programmer productivity with an associated understanding of the reasons for performance loss.
- The parallel programming interface must properly differentiate between, on the one hand, what performance optimizations the compiler can do, and, on the other hand, which ones the programmer must do. A failure to get this right will make it impossible to write efficient code for significant cases. The compiler and language syntax must allow the programmer to be able to optimize the cases that the compiler cannot. In particular, optimizations for data locality that cannot be managed reliably by the compiler must be accessible to the programmer in some form.
- The programming interface should allow portability of application codes across the entire range of applicable platforms.
- It should be possible to run existing legacy application codes with a modicum of performance and, when possible, permit an incremental performance upgrade path for development. As mentioned earlier, reuse of code greatly decreases required development efforts and incidence of defects.
- The programming interface should naturally allow for programming abstractions. On the other hand, it should also allow the writing of code for which the experienced programmer

can “see thorough” the source code to get some idea of the underlying machine operations generated.

- The programming interface or language should permit good software engineering practices (e.g., encapsulation and abstraction, without undue performance loss).

### 5.7.4 Application Readiness

Disruptive changes in hardware technology on the way to attaining exaflop computing promise dramatic shifts in programming methodologies. Changes of this nature will for some time put in flux the more established programming approaches based on MPI and/or OpenMP.

Historically, disruptions such as this have been followed by periods of programming model innovation in which programming principles and best practices emerged from developer communities. For the current situation, these efforts should begin now and be accelerated in order to prepare codes for exaflop computing.

As has been typical in the past, software modernization efforts will commonly begin with small, incremental changes to the source code base along the lines of optimizing kernels and code hot spots. The next step will be to perform a major code refactoring or rewrite of the applications in toto to systematically take advantage of the performance gains offered by the new hardware. This effort may require several phases of prototyping before determining an effective programming methodology.

To the maximum extent possible, code teams must design and write application codes to be independent of foreseeable hardware and software changes. Such an approach manages the risk of not knowing which parallel programming interface will reach final marketplace acceptance. Programming principles such as separation of concerns should be used to separate algorithm from implementation to as great a degree as possible.

Compiler vendors and system software developers must provide robust, well-optimized, stable, standards-based programming interfaces to which code can be written. These tool developers must focus primarily on providing robust basic core- and node-level optimizations, as risk mitigation against the possibility that advanced techniques such as general-purpose automatic parallelization do not become effective.

Hardware and software vendors must continue the routine practice of providing training, support and interaction, to assist application developers in making the needed transitions.

## 5.7.5 Software Optimization

OLCF science application development teams reported code optimization as a bottleneck to effective software development.

A successful computational science application team requires expertise in a range of areas, including physics, software, computational mathematics, computer science, computer operations and computer hardware. The current practice of mandating a division of labor regarding these expertise areas has served the application development process well and should be continued. In particular, code optimization efforts can be delegated to a specialist who has fewer responsibilities in the other technical areas of the project and can focus on the task at hand. At the same time, development team members must to some degree be generalists, to the extent that all team members understand the big picture well enough to be able to mesh together the varied and sometimes conflicting concerns.

The degree of emphasis to be placed on code optimization is a tradeoff between programmer effort, code maintainability and the potential for code performance gains per unit of effort. The programmer productivity regarding code optimization efforts can be enhanced in several ways.

Better profiling tools are needed to make it easier to understand the performance of large-scale parallel applications. These tools should be robust, accurate, cross-platform, cross-language and scalable to millions of threads. They must not only collect data regarding performance effectively but must also report results in a useful way to the developer. Due to the challenge of these goals, favor might be given to more lightweight tools that have strong reliability and leave a small code base footprint compared with heavyweight tools that are prone to lag behind swiftly changing hardware and system software in their support. The FPMPI tool [Gropp and Buschelman 2004] is a good example along these lines.

The practice should continue of vendor trainings to help developers understand how to write optimized code for newer computer hardware and system software. Subject to security concerns, mechanisms such as discussion forums or wikis can be used when appropriate to conserve experiences and best practices in application performance optimization.

As algorithms, compilers and computer hardware have become more complex, the number of tuning options that can be used to improve code performance has increased dramatically, making it impractical to tune codes manually over the entire range of options. Autotuning methods give promise for significant gains along these lines. However, autotuning technology needs to mature to be more easily usable by the general programmer. Compiler hinting holds potential promise as well but must become more production-ready.

## *5.8 Software SQA and V&V Requirements*

Users of HPC science applications and their audience must have assurance of the accuracy and validity of generated science results. Application software quality assurance (SQA) and verification and validation (V&V) are essential components in the life cycle of an application.

SQA is a "systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system product conforms to established functional technical requirements as well as with the managerial requirements of keeping the schedule and operating with budgetary confines." [Galin 2004].

Verification is the process by which one assures that the code "equations are being solved correctly"—namely, through "solution verification", which ensures that the numerical solutions are correct (accurate, convergent) and "code verification", which ensures that the software implementation of the associated algorithms is correct (bug free). Validation is the process by which one assures that the code "equations are correct"—namely that the model formulation accurately and reliably describes reality (matches experimental data).

The growing need for V&V has been described as a "looming crisis in computational science" [Van De Vander et al. 2005]. As software becomes more complex from use of more detailed models and multiple physics packages and hardware becomes more complex from heterogeneity and greater numbers of threads, the need for better SQA and V&V continues to increase. Sources of error in codes are becoming more difficult to find; in this environment, assuring model and software quality is an obvious approach to compensating for the impact of growing hardware and software complexities.

Respondents to the OLCF requirements elicitation identified a variety of procedures currently in use for model and software verification and validation. These include the following:

- Regression testing;
- Comparison with results from other codes;
- Comparison with experiment;
- Comparison with observation;
- Comparison with earlier research as reported in the scientific literature;
- Use of a test suite of established test problems;
- Use of test problems with analytic solutions;

- Comparison with results on different platforms;
- Comparison with results using slightly different inputs (sensitivity analysis);
- Comparison with theory;
- Grid convergence studies; and,
- Comparison with results using different models.

In one research study at a national laboratory, a detailed analysis of major code development projects revealed that V&V and strong software project management were essential to project success [Post and Votta 2005]. However, the same study showed that even in the presence of strong team and institutional commitment, V&V is a very difficult problem.

SQA and V&V efforts for OLCF projects should include the following:

- Code verification efforts should at least include standard testing approaches from the software engineering field, including unit, system and regression testing.
- V&V should be performed on individual components as well as the entire code.
- Stronger institutional support should be given to SQA and V&V efforts. Just as purchasing an "insurance policy" is not a waste of resources but is an integral part of risk mitigation, SQA and V&V are not a waste of project resources but are an important part of assuring science quality.
- The proposal process for being awarded computational resources could set forth specific SQA and V&V requirements as a precondition for funding.
- Projects could be required to disclose specific risks concerning the accuracy and validity of the science generated.
- The broader computational science research community must address at a wider scale the issue of the criteria required to certify that the results of a simulation are valid and thus credible.

## 5.9 Application Usage Workflow Requirements

The way applications are used in practice is the ultimate driver of application requirements. An idealized workflow for performing a single computational science experiment using an OLCF platform consists of these steps: define the problem; build the model (e.g., generate computational grids); validate the model; prepare run inputs; perform preparatory runs as needed; initiate primary run or runs; generate checkpoint/restart and analysis data; restart as needed; postprocess data; generate visualizations; perform analysis; validate results; archive data; and finally, disseminate conclusions.

In practice, a research effort consists of many experiments such as this. These experiments can be performed either sequentially or concurrently, as the specifications for later experiments may or may not depend on the findings of earlier ones. The workflow can be dynamic, as intermediate results of a simulation suggest new avenues of computational experimentation. These relationships embody the actual science production workflow.

The OLCF requirements elicitation process was used to poll domain scientists regarding the details of the workflow for their respective projects. This process revealed a number of commonalities in workflow requirements. The following were some of the findings:

- **Bottlenecks:** The largest perceived workflow bottleneck was the actual execution time or queue wait time of the application jobs on the targeted platform. Naturally, this factor can be addressed directly by procuring larger, more powerful HPC systems. However, additional factors were seen as bottlenecks, such as experiment design, mesh generation, data analysis and off-site data movement.
- **Visualization:** A variety of visualization and analysis tools are used across projects, such as ParaView, VisIt, IDL, MatLab, VMD, Gnuplot, Tecplot, and Ensight.
- **Archived data:** The amount of simulation data archived ranged from a small fraction to the entirety of the simulation data.
- **Anticipated future changes:** The primary anticipated workflow change going forward is an increase in the scope and accuracy of the science, leading to greater demands on computation and I/O storage requirements. An additional change expected by some projects is greater use of on-site or even application code in-situ analysis and visualization techniques, to deal with the problem of managing growing quantities of data.

The following are identified as key requirements for workflow management on next-generation leadership computing hardware.

1. **Storage and manipulation of growing amounts of data.** Higher fidelity simulations will generate exponentially increasing amounts of data. This growth encompasses both the aggregate amount of data and the number of files, stressing both data and metadata storage. Science workflow is limited by both the size and speed of data storage systems. Future

systems must address the need for large, fast storage under the constraints of a multi-user environment.

2. **Data resilience and integrity.** The presence of more data and metadata increases the probability of failure. Fault tolerance applies not only to processors and DRAM memories but also to disk storage. Storage systems, file formats and I/O libraries must be robust and fault tolerant to handle increasing quantities of data.

3. **Automation of workflow and data management.** As science models become more complex, science codes are coupled to produce new science, core counts are increased, the volume of data produced grows, and the human factor in managing science workflow becomes increasingly stressed. The science process workflow must be automated as much as possible to manage this growth in complexity. Workflow management tools can be of use in this regard (e.g., see [Cummings et al. 2008]).

4. **Data organization to enable analytics.** Once created, science data must be available to analyze, with fast response times for scientists analyzing the data. As the quantity of data grows, reaching this objective becomes more challenging. Data management systems must provide tools to control the layout of data on mass storage devices to facilitate fast queries. In situ analysis when appropriate is a further step to lessening the data access burden. Furthermore, advanced mathematical algorithms must be provided to support operations such as dimension reduction, statistical analysis and feature identification.

## *5.10 Data Management Requirements*

A productive simulation environment requires systems and tools capable of storing, transmitting, and manipulating extremely large datasets, both within and across multiple HPC centers. Large-scale simulation platforms are capable of generating multiple petabytes of data per year, and archival data storage requirements are growing exponentially as illustrated in Fig. 5.3.

In addition to archival storage, parallel I/O storage data growth is increasingly challenging. As illustrated in Fig. 5.4, data stored in the OLCF parallel I/O environment recently increased by over 800TB in just three months. As simulation environments continue to scale both in terms of compute cores and system memory, the need to store massive amounts of data for both defensive and productive I/O increases dramatically. Managing this dramatic storage growth requires careful planning of system upgrades and improved information lifecycle management.

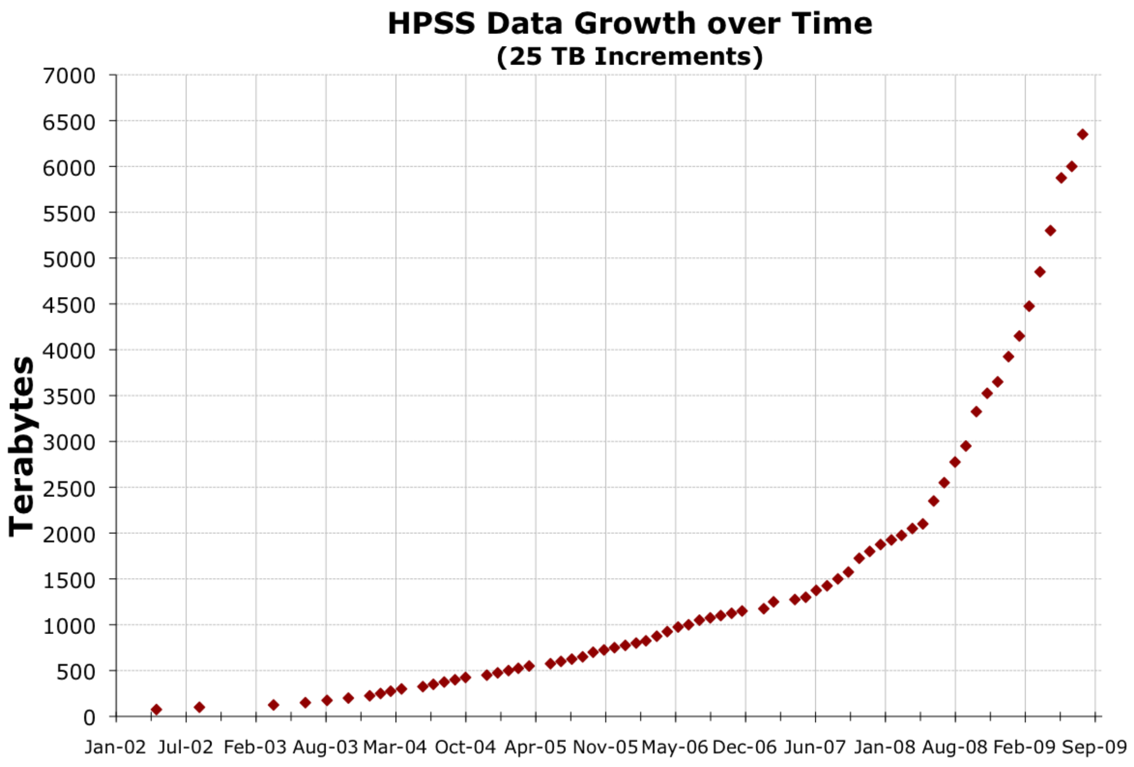


**Fig. 5.3. OLCF archival storage growth**

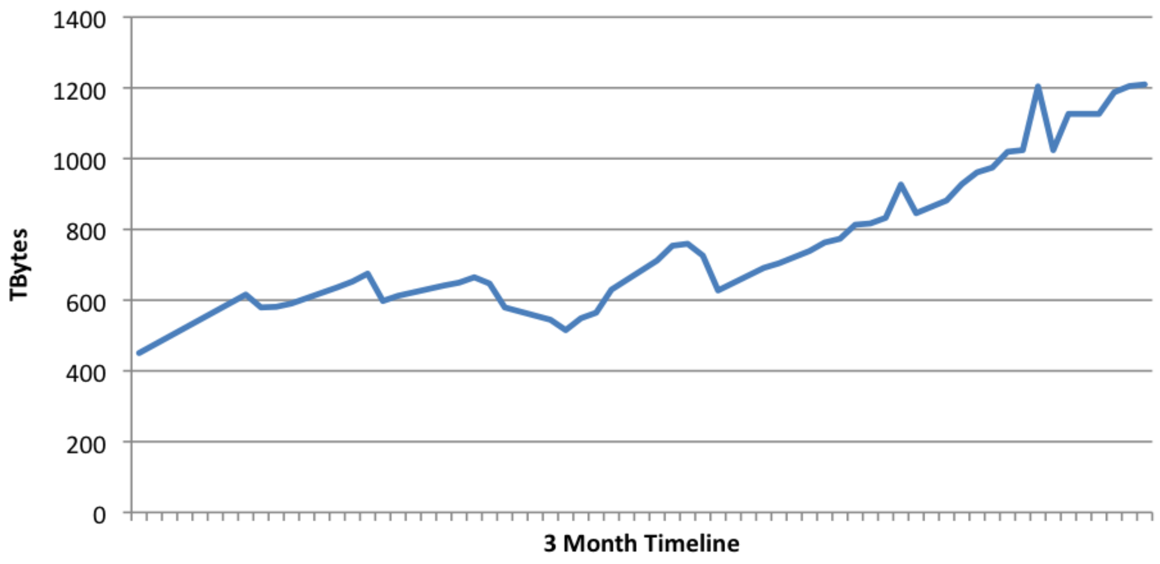


**Fig. 5.4. Parallel I/O environment data growth**

As the number of datasets located both in archival and in online storage increases, the ability to effectively manage, identify and retrieve this data becomes increasingly challenging. With over 150 million files in online storage today and projections of over 1 billion files in the near future, parallel tools must be developed to manage these datasets. Standard system tools for data management such as *cp*, *tar*, and *find* must be parallelized to take advantage of parallel I/O environments and high performance archival systems.

The identification of a handful of relevant datasets among over 1 billion potential candidates will require improved metadata tagging and search capabilities. Standardization of metadata and improved metadata extraction will allow users of these scientific datasets to quickly identify datasets of interest while providing a basis for automation of a number of data-analysis activities. To ensure the integrity of these scientific results, the storage environment will need to be enhanced to provide data provenance features allowing users to determine both the origin and subsequent transformations of these datasets.

As science teams increasingly utilize multiple compute centers and share data among these centers, high-performance data transfer mechanisms between these sites are of increasing importance. Current generation networking technologies limit data transfers to a maximum of 1.25 GB/sec over a single 10Gb WAN link. As an example, transferring a 300TB dataset at these rates could take days or more to complete in the absence of network congestion, unlikely on a shared network such as ESnet. The Advanced Networking Initiative (ANI) which will provide a 100 Gb native optical network loop among the OLCF, ALCF, and other facilities will be required to allow reasonable transfer times for these extremely large datasets and allow effective use of multiple computational resources spanning a number of HPC centers.

# 6. CONCLUSIONS AND RECOMMENDATIONS

The purpose of the OLCF requirements modeling process is to insure that institutional resources are effectively procured, deployed and managed so that nationally-identified science objectives are attained cost effectively and with a low risk factor.

A major finding of the OLCF 2009 requirements process is that the next stages of advance in high-end computational science are likely to be significantly more challenging than past efforts. Certainly, the challenges of high performance computing have continuously increased over time, as hardware, science models, algorithms and application codes have increased in complexity. Furthermore, adapting to periods of disruption in hardware architecture has been a great challenge for application software.

However, the current shift to heterogeneous processors may be the most significant challenge yet, since (1) the performance penalty for not modifying applications to take advantage of new hardware is the highest since the advent of message passing programming, and (2) HPC science applications are more complex than ever before and thus increasingly difficult to revise for new hardware.

Planning for innovation is an inexact science. Managing requirements in a period of change has a level of risk and uncertainty. However, the OLCF requirements process has successfully identified actionable steps to prepare for next-generation leadership science. The following are the primary findings and recommendations from this process:

- Next-generation science models will require at least an order of magnitude more compute power than current hardware provides. The next OLCF system will require a capability of 10-20 petaflops or more.
- Science applications make use of multiple computational algorithm “motifs”. The motifs tend to be shared across multiple science domains and applications. The reliance on certain motifs is increasing over time. It is recommended that effective libraries be developed to support more of these computational motifs. Such libraries would be leveraged to provide improved performance and productivity across science domains and would also free developers to focus on more pressing concerns.
- More research and deployment of improved algorithms are needed to meet the challenge of increased numbers of processing threads and growing demands for data locality.
- MPI is the predominant means of obtaining parallelism. Use of OpenMP and threads is increasing. Programming interfaces such as CUDA, OpenCL and new directives-based methods will be required for new hardware. These interfaces must be usable and provide

access to the performance potential of the new hardware. Application teams must prepare for these changes.

- Applications will require varying levels of refactoring to adapt to heterogeneous processor hardware, from rewriting of kernels to major code restructuring.
- Applications on average spend one third of runtime in scientific or mathematical libraries. For future hardware, libraries must be well-tuned to system hardware for targeted use cases.
- The system software stack must provide a diverse set of tools and services to support parallel applications.
- Application performance is particularly dependent on several specific hardware attributes, including interconnect bandwidth, node peak flops, memory bandwidth, MTTI, node memory capacity, memory latency and interconnect latency. Future platforms cannot afford to neglect any of these attributes in the pursuit of effective leadership science.
- Processor accelerators such as NVIDIA general purpose GPUs directly address requirements for three of the key hardware attributes, including node peak flops, memory bandwidth, and memory latency.
- Better software tools must be made available to help manage the increasing complexity of HPC hardware and software. Better tools can be obtained by increasing support of tool developers, by supporting the construction of more lightweight, easily-maintained tools or by leveraging more heavily existing COTS tools.
- To address concerns of software quality and robustness, application developers must incorporate proven software development techniques into project workflows. Institutional support must be given for this. V&V efforts must be strongly supported.
- Increasing science accuracy will drive requirements for greater amounts of secondary storage. Capabilities for storing, transferring and visualizing larger quantities of data will be required.

The challenges of effectively using newly deployed computing hardware are daunting. To address nationally mandated computational science goals, efforts must be undertaken now to prepare for and adapt to changes in leading-edge HPC computing hardware.

# ACKNOWLEDGEMENTS

The authors would like to thank Thomas Zacharia and Jeff Nichols for their visionary leadership in chartering and founding the NCCS OLCF; Buddy Bland and the rest of the NCCS management for their leadership and direction of the NCCS OLCF Project and day-to-day operations; the support staff and management of the DOE Office of Science ASCR Program Office (Michael Strayer, Daniel Hitchcock, Walt Polansky, Barbara Helland, and Fred Johnson); members of the ASCAC subpanel (in particular, chair Gordon Bell) on science-based performance metrics for ASCR computational facilities; and finally the principal investigators and project team members on each of the NCCS OLCF and INCITE Projects.

The authors thank Sarp Oral and Galen Shipman for their contributions to the I/O and Data Management sections of the document. The authors also thank Richard Graham, Rebecca Hartmann-Baker, Ricky Kendall, Scott Klasky, and other members of the NCCS for their valuable comments and contributions. The authors also thank the principal investigators, science application team members and liaisons for their valuable input.

The staff in the NCCS, particularly members of the Scientific Computing Group, would like to thank the computational scientists involved in the 2009 INCITE projects that received allocations on the OLCF. Without their applications, science achievements, time, and support of the OLCF, this facility would not have a reason to exist.

This work is supported by the Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract DE-AC0500OR22725.

# REFERENCES

Agullo, Emmanuel, Jim Demmel, Jack Dongarra, Bilel Hadri, Jakub Kurzak, Julien Langou, Hatem Ltaief, Piotr Luszczek, and Stanimire Tomov. 2009. "Numerical Linear Algebra on Emerging Architectures: The PLASMA and MAGMA Projects," *Journal of Physics: Conference Series* **180,** 012037 (http://stacks.iop.org/1742-6596/180/012037).

Asanovic, Krste, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. 2006. *The Landscape of Parallel Computing Research: A View from Berkeley*, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley (http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html).

ASCR. 2009. *Advanced Scientific Computing Research Funding Profile by Subprogram, 2009*, Department of Energy, Office of Science (http://www.er.doe.gov/obp/FY_09_Budget/ASCR.pdf).

Bailey, F. Ronald, Gordon Bell, John Blondin, John Connolly, David Dean, Peter Freeman, James Hack, Steven Pieper, Douglass Post, and Steven Wolff. 2007. *Petascale Metrics Report*, Advanced Scientific Computing Advisory Committee, Department of Energy, Office of Science (http://www.er.doe.gov/ASCR/ASCAC/Reports/PetascaleMetricsReport.pdf ).

Colella, P. 2004. *"*Defining Software Requirements for Scientific Computing*,"* Defense Advanced Research Projects Agency HPCS presentation.

Collette, Michael, Bob Corey, and John Johnson. 2004. *High Performance Tools and Technologies*, Computer Applications and Research Department, Lawrence Livermore National Laboratory, Livermore, Calif. (https://computing.llnl.gov/tutorials/performance_tools/HighPerformanceToolsTechnologiesLC.pdf).

Cummings, J., A. Pankin, N. Podhorszki, G. Park, S. Ku, R. Barreto, S. Klasky, C. S. Chang, H. Strauss, L. Sugiyama, P. Snyder, D. Pearlstein, B. Ludascher, G. Bateman, A. Kritz, and the CPES Team. 2008. "Plasma Edge Kinetic-MHD Modeling in Tokamaks Using Kepler Workflow for Code Coupling, Data Management and Visualization," *Communications in Computational Physics* **4**(3), 675–702 (http://www.global-sci.com/freedownload/v4_675.pdf).

Demmel, James, Mark Hoemmen, Marghoob Mohiyuddin, and Katherine Yelick. 2007. *Avoiding Communication in Computing Krylov Subspaces*, Technical Report UCB/EECS-2007-123, University of California, Berkeley (http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-123.html).

Galin, D. 2004. *Software Quality Assurance—From Theory to Implementation*, Pearson Education Limited, Harlow, U.K.

Geist, Al, Phil Colella, and Mike Heroux. 2007. *Workshop on CS/Math Institutes and High Risk / High Payoff Technologies for Applications, Final Report*, U.S. Department of Energy, (http://www.er.doe.gov/ascr/ProgramDocuments/Docs/MathCSWorkshopReport.pdf).

Grady, Robert B. 1992. *Practical Software Metrics for Project Management and Process Improvement*, Prentice Hall, Upper Saddle River, N.J.

Graham, Susan L., Marc Snir, and Cynthia A. Patterson, eds. 2005. *Getting up to Speed: The Future of Supercomputing*, National Academies Press, Washington, D.C. (http://www.nap.edu/openbook.php?isbn=0309095026).

Gropp, William, and Kristopher Buschelman. 2004. *FPMPI, Fast Profiling Library for MPI*, Argonne National Laboratory, Argonne, Ill. (http://www.mcs.anl.gov/fpmpi).

HPCS DTWG. 2007. HPCS Development Time Working Group, Conference Publications List, University of Maryland, College Park (http://hpcs.cs.umd.edu/index.php?id=14).

IBM. 2009. "Rational Purify Product Line," International Business Machines Corp. (http://www-01.ibm.com/software/awdtools/purify).

Joubert, Wayne D., and Graham F. Carey. 1992. "Parallelizable Restarted Iterative Methods for Nonsymmetric Linear Systems. Part I: Theory, Part II: Implementation," *International Journal of Computer Mathematics* **44**(1–4), 243–290 (http://www.informaworld.com/openurl?genre=article&issn=0020-7160&volume=44&issue=1&spage=243).

Kandt, Ronald Kirk. 2006. *Software Engineering Quality Practices*. Auerbach Publications, Boca Raton, Fla.

Kasim, Henry, Verdi March, Rita Zhang, and Simon See. 2008. "Survey on Parallel Programming Model," NPC '08: Proceedings of the IFIP International Conference on Network and Parallel Computing, Shanghai, China, *Lecture Notes in Computer Science* **5245,** 266–275 Springer (http://www.springerlink.com/content/u61n6x07u7j26x73).

Kogge, Peter, ed. 2008. *ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems*, Defense Advanced Research Projects Agency (http://users.ece.gatech.edu/~mrichard/ExascaleComputingStudyReports/ECS_reports.htm).

Kothe, Douglas, and Ricky Kendall. 2007. *Computational Science Requirements for Leadership Computing*, Report ORNL/TM-2007/44, National Center for Computational Sciences, Oak Ridge National Laboratory (http://www.nccs.gov/wp-content/media/nccs_reports/ORNL_TM-2007_44.pdf).

Lawson, C. L., R. J. Hanson, D. R. Kincaid, and F. T. Krogh. 1979. “Basic Linear Algebra Subprograms for Fortran Usage,” *ACM Trans. Math. Softw.* **5,** 308–323 (http://doi.acm.org/10.1145/355841.355847).

Mucci, Phil, et al. 2009. “PAPI, Performance Application Programming Interface,” Innovative Computing Laboratory, University of Tennessee, Knoxville (http://icl.cs.utk.edu/papi).

Nielsen, Jakob. 1993. *Usability Engineering*. Morgan Kaufmann, San Francisco.

Post, Douglass E., and Lawrence G. Votta. 2005. “Computational Science Demands a New Paradigm,” *Physics Today* **58**(11), 35–41, American Institute of Physics.

Raymond, Eric S. 1999. *The Cathedral and the Bazaar*, O'Reilly Media (http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar).

Tomov, Stanimire, Jack Dongarra, and Marc Baboulin. 2008. *Towards Dense Linear Algebra for Hybrid GPU Accelerated Manycore Systems*, Technical Report UT-CS-08-632, University of Tennessee, LAPACK Working Note 210 (http://www.netlib.org/lapack/lawnspdf/lawn210.pdf).

Van De Vanter, Michael L., D. E. Post, and Mary E. Zosel. 2005. “HPC Needs a Tool Strategy,” pp. 55–59 in *SE-HPCS '05: Proceedings of the Second International Workshop on Software Engineering for High Performance Computing System Applications*, St. Louis, Missouri, 2005, ACM, New York (http://doi.acm.org/10.1145/1145319.1145335).

Vetter, Jeffrey, ed. 2007. *Workshop on Software Development Tools for Petascale Computing*, Washington, D.C., August 1–2, 2007 (http://www.er.doe.gov/ascr/WorkshopsConferences/Docs/sdtpc_workshop_report.pdf).

Williams, Sam, D. A. Patterson, and A. Waterman. 2009. “Roofline: An Insightful Visual Performance Model for Multicore Architectures,” *Communications of the ACM* **52,** 65–76 (http://doi.acm.org/10.1145/1498765.1498785).

# APPENDIX A. OLCF OVERVIEW

The U.S. Department of Energy-funded Oak Ridge Leadership Computing Facility (OLCF) is part of the National Center for Computational Sciences (NCCS) at Oak Ridge National Laboratory (ORNL).

The NCCS was established in 1992. The mission of the NCCS is to advance the state of the art in high performance computing and to make available the capabilities of high-end parallel supercomputers to scientists in a wide variety of computational science domains. NCCS works with academia, industry, laboratories and government agencies to provide a computational environment that enables the scientific community to employ this computational capability to solve important problems in diverse areas such as fusion, climate, astrophysics, materials science, biology, nanoscience, chemistry, accelerator physics, combustion, engineering, and other disciplines vital to maintaining U.S. science leadership.

The NCCS leadership computing efforts initially focused on a Cray X1 computer, "Phoenix," with peak speed of 6.4 teraflops and a Cray XT3 system, "Jaguar", with peak speed of 26 teraflops. A series of upgrades resulted in the Cray XT4 "Jaguar" system in 2007 becoming the second fastest computer in the world, achieving 101.7 teraflops on the LINPACK benchmark and 119 teraflops peak speed. In 2008, the NCCS began operation of the Cray XT5 "Jaguar" system, the world's fastest computer for open science, which together with the Jaguar XT4 system delivered 1.64 petaflops of performance. In late 2009 the Jaguar XT5 system was upgraded to a capability of 2.595 petaflops peak speed.

The Jaguar XT5 system has been the winner of numerous awards in high performance computing. The 2009 Gordon Bell Prize winner—a team led by Markus Eisenbach of ORNL—achieved 1.84 petaflops on Jaguar with a scientific simulation of high-temperature superconducting materials. Jaguar took first place in three of four competitions at the 2009 High-Performance Computing Challenge (speed in solving a dense system of linear equations, Global-Fast Fourier Transform and sustainable memory bandwidth) and third place in the fourth category (speed in executing Global-Random Access).

In addition to Jaguar, the OLCF houses an array of support systems and services, including high-performance memory, file systems, archival storage, networking capabilities and visualization hardware including the EVEREST visualization system.

The NCCS facility includes a diversity of support personnel, including the Scientific Computing Group (SciComp), whose role is to collaborate with OLCF users to accelerate scientific progress. SciComp members apply their extensive experience in parallel algorithm development and implementation, model formation, software development, and code porting and tuning to support science application development teams.

In 2010, Oak Ridge National Laboratory will make nearly 950 million core-hours available on Jaguar under the DOE's Innovative and Novel Computational Impact on Theory and Experiment, or INCITE, program. The INCITE program awards allocations on some of the world's largest computers to address grand challenges in science and engineering.

To design the next-generation HPC system, NCCS is working in collaboration with the DOE Office of Science and the Department of Defense High Productivity Computer System (HPCS) program. By 2011-2012, the Office of Science plans to install a computing platform of roughly 20 petaflop capability resulting from the HPCS program. Further into the future, the plan is to install a 100-250 petaflop platform in the 2015 time frame and an exaflop machine by 2018.

# APPENDIX B. REQUIREMENTS ELICITATION

Members of the Scientific Computing Group at the Oak Ridge Leadership Computing Facility (OLCF), part of the National Center for Computational Sciences (NCCS) at Oak Ridge National Laboratory (ORNL), surveyed numerous scientists in a broad range of scientific domains and asked them to speculate on requirements for their scientific application(s) on Leadership Computing platforms in the next 3–5 years. A large fraction of the information, guidance, and plans outlined in this document is derived from the answers provided in these surveys from this expert community of leading computational scientists. Without their insight, knowledge, and experience, the application requirements outlined in this document would not have nearly the fidelity or significance. The survey questions are listed here.

**Project Overview**

- Project name
- Contact information for the project (principal investigators, emails, phones, URL)
- Scientific domain (chemistry, fusion, high energy, nuclear, other)
- Team size
- Team institutional affiliation(s)
- Composition of team personnel by training and by chief technical or scientific focus area
- Team resource allocation for code development, code maintenance, code utilization
- Code(s) used and/or developed
- Description of each code, including hyperlinks to home page
- Development history (including where and who)

**Science Motivation and Impact**

- Why does your science need leadership computing (use of a large fraction of a large-scale HPC system)?
- Without leadership computing, can progress be made at all? Or as fast?
- What science questions are you answering?
- What impact will your answers have on your field? Other fields?
- What do you consider to be the most exciting and important aspects of this research?
- How will you use your results to confirm observations or measurements (e.g., are you simulating a particular experimental device, or will your findings be tested in other ways)?
- Will your models have a predictive capability?

**Application Models**

- Briefly describe the basic physics or system being modeled by your code.
- How do you envision your model(s) for science changing in the next several years as more advanced computer resources become available?

- What is the application area (molecular physics, nanoscale science, climate, environment, combustion, fusion, nuclear energy, biology, chemistry, astrophysics, nuclear physics, accelerator physics, QCD, aerodynamics, etc.)?
- Are your models deterministic? Stochastic? Both? If deterministic, how are your models expressed (e.g., partial differential equations)?
- Is the model steady or dynamic (e.g., boundary value problem or initial value problem)?
- Are multiple, simultaneous physical processes modeled (i.e., multiphysics
- How many independent variables or degrees of freedom per discrete solution point currently describe your physical system?
- Does the nature of the model vary over space and/or time (e.g., more DOFs or different connectivity in different physical regions)?
- How are these factors expected to change in the next several years as models are improved and more advanced computer resources become available? What new models or capabilities would be of chief importance to develop for use in improving computer hardware?

## Application Algorithms

- What types of algorithms and computational mathematics are used (e.g., structured grids, unstructured grids, adaptive mesh refinement, spectral/FFT, dense or sparse linear algebra, Monte Carlo methods, finite state machines, combinational methods, graph traversal, dynamic programming, particles, backtrack/branch and bound, graphical model inference, finding nearest neighbors)?
- How are these factors expected to change over the next several years?
- If dense or sparse linear algebra, what methods are used?
- If grids, what types of method are used (finite difference, finite element, etc.) (Eulerian, Lagrangian, MD)?
- If AMR, what type of refinement is used (patch-based, cell-based, etc.)?
- Do your algorithms adaptively change as a function of space and/or time based on the data?
- Have you been able to quantify convergence properties and numerical errors of your algorithms?
- What is the largest source of numerical error in your solution algorithms?
- How are these factors expected to change over the next several years, as new algorithms are implemented and more advanced computer resources become available? What new algorithms would be of chief importance to develop for use in improving computer hardware?
- How are these factors expected to change over the next several years as new algorithms are implemented and more advanced computer resources become available? What new algorithms would be of chief importance to develop for use in improving computer hardware?

## Application Parallelization Strategy

- What is your current data decomposition model (e.g., distributed, domain replicated)? What is the primary axis of parallelism (e.g., space, time, task)? Are you instantiating parallelism with MPI tasks, threads, or both?
- Do your science requirements dictate that your code scale in a strong sense (fixed problem size, increase concurrency), a weak sense (fixed problem size per node), or both? What parameter do you scale on?
- Is your application load balanced?
- What method is used for load balancing?
- Does your application require dynamic repartitioning
- What techniques, if any, are used to increase memory locality for improved cache utilization?
- What is your current I/O model (e.g., parallel, serial through a single PE, hybrid)?

- How important is a fast underlying message-passing fabric to obtaining optimal performance from your code?
- Is your code sensitive to network latency, bandwidth, or both?
- Are your algorithms sensitive to on-node memory bandwidth?
- How are these factors expected to change over the next several years?
- Is the domain of dependence for any given state variable local (i.e., dependent upon other nearby state variables) or global?
- What new parallelization, cache optimization, or I/O strategies are being considered?
- Does your application have unexploited concurrency?

## Application Software

- What platforms does your code run on? What is your preferred platform?
- How many lines of code comprise the primary models you plan to run (single lines of code, function points, etc.)?
- What computing and/or scripting languages are employed (Fortran, C, C++, Python, etc.)?
- What libraries do your applications require, including I/O libraries such as NetCDF or HDF?
- How are these factors expected to change over the next several years?
- What compiler vendors are supported and typically used (PGI, PathScale, etc.)?
- Are there functionalities in your code that could be offloaded to a library function?
- To what extent does your team develop and use its own codes? Codes developed by others in the DOE and general scientific community?
- What software is used to achieve parallelism (MPI, OpenMP, PThreads, etc.)?
- What software, if any, is used for I/O?
- What data file formats are used (HDF5, NetCDF, PHDF5, pNetCDF, etc.)?
- What analysis programs (data mining, visualization, etc.) do you run? (IDL, Matlab, VisIt, EnSight, AVS, Gnuplot, SCIRun, Python/Perl scripts, R, etc.)?
- Can your application execute on a heterogeneous platform (i.e., nodes having different hardware)?

## Application Development Process

- For the primary model(s) you plan to run, how often is a new software release issued?
- What software development tools are used?
- What is the biggest time bottleneck in the software development cycle?
- Is the software you plan to run under active development?
- How are these factors expected to change over the next several years?
- Is there any structure imposed on the development of your software product (e.g., do you use software life-cycle model such as waterfall, evolutionary delivery, etc.)?
- Please list the specific tools or processes used for the following software engineering practices:
    - Configuration management (revision control, etc.)
    - Quality control and testing
    - Documentation
    - Bug and issue reporting and tracking
    - Code reviews
    - Project planning
    - Project scheduling and tracking
    - Collaboration management
- What build system do you use? Is it "maintenance free"?
- Does your code have a centralized software repository?

- What is the split between code development on computer center computers and code development on computers at other institutions?
- Is your code developed by a single individual or by a team?
- What is the maturity of your software in terms of age, testing breadth and depth, and ability to model the science problems planned?

## Application Software Quality Assurance (SQA) and Verification and Validation (V&V)

- When moving your codes to ORNL resources, how do you plan to verify that the model(s) are behaving as expected?
- What kind of testing (e.g., unit, regression, integral) do you perform? How many of each?
- What is your solution validation strategy?
- What is your verification strategy? How are your simulations verified (solving equations correctly)?
- What is your validation (solving the right equations) strategy? What experimental facilities do you use for validation? Does your project have adequate resources for validation?
- 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 (i.e., have the highest uncertainties or sensitivities)?
- What V&V tools and methodologies (e.g., method of manufactured solutions) do you use?
- How are these factors expected to change over the next several years?

## Application Usage Workflow

- Describe a typical use case for the code(s) you will run. An example would typically include problem definition, problem setup, main compute phase, postprocessing, data analysis and visualization, and dissemination of results.
- What are the time-intensive bottlenecks for the use case outlined above?
- What visualization and analysis tools do you typically use (e.g. NCO, Ferret, VisIt, IDL)?
- What maximum simulation turnaround time can you tolerate and still move your science forward?
- For a given processor count, what fraction of your simulation run time is spent in I/O?
- What is the expected annual use of resources in terms of processor-hours, disk, and archival storage?
- What is the size of a typical job in terms of core count, memory, disk, archival storage, and wallclock time?
- What are your temporary and archival storage size needs for analysis dumps and restart dumps (expressed as a function of the simulation core count)?
- Do you archive all of your data from a simulation, or just a fraction of it?
- What size are the external communities your code or datasets support?
- What is the number of users?
- What proportion of your output data do you transfer to your home institution?
- How are these factors expected to change over the next several years?
- Are your computational experiments sequential (i.e., the current dependent upon the previous result)?
- What turnaround time would allow detailed parameter studies and optimization?
- How much time do you spend analyzing current runs? From the end of the simulation to the time you publish?

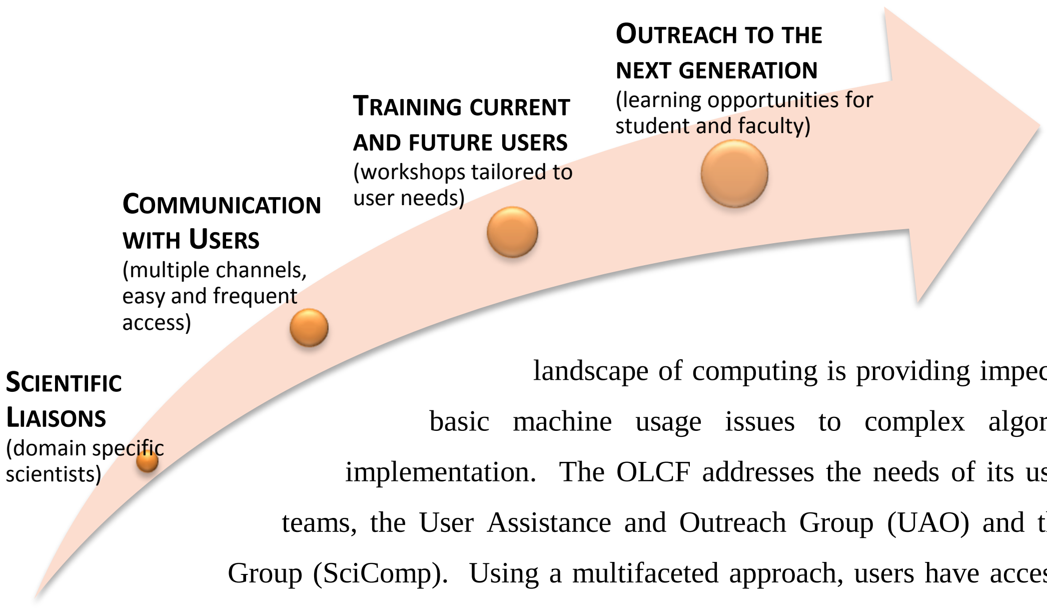
- What are the frequency and size (in terms of fraction of simulation image) of your restart and graphics dumps?
- What is the steady-state use of resources on a production basis per month, in terms of processor-hours, disk, and tertiary storage rate of change?
- Does your analysis program read in all of your data (all of the variables, all of the timesteps
- What is your current bottleneck in analysis (e.g., data movement, coming up with new routines for new analysis, trying to decide which routine will work best with a new dataset, comparing results to older data and experimental data)?
- What is the split between interactive and batch use?
- How do you monitor your running simulation? (ASCII output and run gnuplot? Transfer file over to your cluster and run another program? Real-time monitoring? Visualization tool on restart dumps?)
- How do you use archival storage? Are your simulation datasets analyzed and used by many others, or are they for single-user backup?
- Based on past experience, what do you anticipate increased computing capabilities will provide?
    - 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 are these factors expected to change over the next several years?

## Application Performance

- Do you have a normalized performance metric for your application (e.g., grind time)? If so, what is it? Is it being tracked?
- What is the maximum demonstrated scalability of the code (in a weak and/or strong sense)?
- What is the maximum projected scalability of the code? How was this figure obtained?
- What is the greatest hindrance to scalability at large core counts? What algorithms are primarily responsible?
- Does your application have a few identifiable performance bottlenecks?
- Are these bottlenecks localized in software?
- What do these parts of the code do?
- Is it likely that significant performance improvements could be attained by implementing further code optimizations?
- Using alternative compilers, compilation options, or libraries?
- What fraction of your simulation run time is spent in communication?
- How much memory per core does your code typically require?
- What fraction of machine peak speed does the code typically attain? What is the primary limiting factor (memory bandwidth, communication bandwidth, load imbalance, I/O, etc.)?
- If hardware flop rate or memory capacity was doubled, how would you make use of these added resources?
- Communication questions
    - For the primary interprocess communication tasks of your code, what is the typical number of other processes communicated with by each process? Does the communication topology vary over space and/or time?
    - What is the distribution of message sizes?
    - What kind of message passing calls does you code use (blocking, asynchronous).
- Memory questions

  - 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?
  - Does your application require an extensive amount of indirect addressing? What is the aggregate computational intensity (flops per memory reference)?
- Computation questions
  - What fraction of the total cycles is devoted to floating point ops, integer ops, logical ops, data movement, etc.?
  - What is the “efficiency” of the code, and how is it measured?
- What-if questions
  - How might the quality (fidelity of physics models) of your science change with platform peak speed and aggregate memory?
  - How might the productivity of your science output change with platform peak speed and aggregate memory?

# APPENDIX C. USER SUPPORT

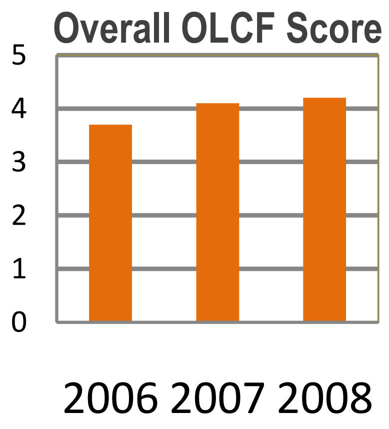

SCIENTIFIC LIAISONS
(domain specific scientists)
COMMUNICATION WITH USERS
(multiple channels, easy and frequent access)
TRAINING CURRENT AND FUTURE USERS
(workshops tailored to user needs)
OUTREACH TO THE NEXT GENERATION
(learning opportunities for student and faculty)

One of the key components to bridging the gap between application scientists and the quickly changing landscape of computing is providing impeccable user support, from basic machine usage issues to complex algorithm development and implementation. The OLCF addresses the needs of its users through two support teams, the User Assistance and Outreach Group (UAO) and the Scientific Computing Group (SciComp). Using a multifaceted approach, users have access to the level of support and features necessary for projects success.

User surveys conducted by an independent third party of OLCF users with 48% response rate from 226 surveys showed:

- ✓ *Overall result in 2008 was 4.2 on a scale from 1 to 5 where 5 is very satisfied*
- ✓ *Every question scored 3.5 (satisfactory) or higher*

User surveys, as well as our frequent and close interaction with users, enable us to see problems immediately and continually improve our facility.



The OLCF provides experts in user support, including Ph.D.-level liaisons from science fields who are also experts in developing and optimizing code for the OLCF systems. Large projects are assigned primary, secondary, and visualization liaisons to maximize opportunities for success on the leadership computing resources. Liaisons address user needs in scientific computing, visualization, end-to-end workflow, and runtime performance.

## Scientific Liaison

A critical area in which the OLCF excels is in assisting the user communities in porting, tuning, and scaling applications to run on these highly scalable systems. The Scientific Computing Group within the OLCF provides a liaison to each project to assist in these activities. Our customers have told us that without the liaisons, using these large systems would be much more difficult and less productive. As we begin deploying accelerator-based computer systems, this activity will make the difference between effectively using the systems and not. Already, multiple members of the group are developing algorithms for accelerators and heterogeneous processor technology.

Duties include:

- Providing guidance and experience to the project team
- Improving performance and scalability of the project application software
- Assisting in redesigning, developing, and implementing strategies that increase effective use of OLCF resources
    - Scalable algorithmic choices and library-based solutions
- Assisting in the planning of new code and algorithm development
- Providing an advocacy interface to the OLCF resource decision entities
    - Resource Utilization Council
    - Technology Council
    - Software Council

## Visualization Liaison

Visualization is an integral component to scientific computing; allowing scientists to delve into their data as well as immediately convey their science to a larger audience. The OLCF provides a visualization liaison to every INCITE project for their post-analysis data processing needs. Support services include:

- Support of visualization tools
- Conversion of data
- Statistical analyses
- Production of publication-ready images
- Production of movies and animations
- Highlighting of science successes to visitors
- Research of new data exploration techniques
- Writing of custom visualization tools and algorithms
- Parallel data analysis support
- Large display support

# APPENDIX D. SURVEY OF APPLICATIONS

The applications listed here are a sampling of the leadership-class applications ported to the Jaguar system. These codes possess high potential for achieving breakthrough science results. The codes span many domains of science and a wide variety of models, algorithms, and software that collectively stress all aspects of a leadership-class computational resource. These applications originate from many different institutions. For each code in this list the following are summarized: physical models, numerical algorithms, scaling performance, and functional software requirements (system software and mathematical libraries). The code data is based in large part on details graciously provided by the relevant code authors and subject matter experts. A list of codes is given in Table D.1. Details for each code are provided in the text following the table.

**Table D.1. Representative applications ported to the Jaguar system**

| Science Category | Research Area | Code(s) |
|---|---|---|
| **Biology** | Biophysics | GROMACS, LAMMPS, NAMD |
| **Chemistry** | Chemistry | MADNESS, NWChem, CP2K, QMCPACK |
| **Earth Science** | Climate | CAM |
| | Geosciences | PFLOTRAN |
| | Ocean Modeling | POP |
| **Engineering** | Combustion | S3D |
| **Fusion** | Fusion Energy | AORSA, GTC, TGYRO |
| **Materials** | Material Sciences | DCA++, gWL-LSMS, VASP |
| | Nanoelectronics | OMEN |
| | Nanosciences | LS3DF |
| **Nuclear Energy** | Neutron Transport | Denovo, UNIC |
| **Physics** | Astrophysics | Chimera, FLASH |
| | Condensed Matter | CASINO |
| | Lattice Gauge Theory | MILC/Chroma |
| | Nuclear Physics | MFDn, NUCCOR |

## D.1 Biology

### Biophysics: GROMACS

*Molecular dynamics simulator*

**http://www.gromacs.org**

GROMACS is a versatile package for performing molecular dynamics, i.e., simulating the Newtonian equations of motion for systems with hundreds to millions of particles. It is primarily designed for biochemical molecules like proteins, lipids, and nucleic acids with many complicated bonded interactions, but since GROMACS is extremely fast at calculating the nonbonded interactions (that usually dominate simulations), many groups are also using it for research on nonbiological systems (e.g., polymers).

| | | | | |
|---|---|---|---|---|
| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
| | C, F90 | MPI | None | None |

| | | | |
|---|---|---|---|
| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** |
| | FFT | fftw_create_plan<br>fftw<br>fftw_destroy_plan | Creates/destroys an object with information required to compute FFT in the FFTW library |

**Algorithms:** Ordinary Differential Equations of Newton's Dynamic; PME, PPPM, Ewald summation

**Scaling:** A representative breakdown of CPU cost for a time step is 85% for force computation, 10% for neighbor finding, and 5% includes time integration, application of boundary conditions, etc. Typically, for biomolecular systems, the force computation is dominated by short-range pairwise interactions and long-range Coulomb interactions. The traditional method for solving the long-range Coulomb part are Ewald summations with the solution to the smooth summation part accomplished via distributed 3-D FFTs on a grid to which particle charge density is interpolated. The 3-D FFT's have a scale computationally O(NlogN) but the communication overhead of the distributed 3-D FFT's make them scale poorly with system size – several all-to-all communication patterns. If we ignore the cost of FFTs (which typically only require >50% of the force computation time for large systems), classical MD simulations scale as O(N) in both memory and CPU cost, where N is the number of particles simulated.traditional method for solving the long-range Coulomb part are Ewald summations with the solution to the smooth summation part accomplished via distributed 3-D FFTs on a grid to which particle charge density is interpolated. The 3-D FFT's have a scale computationally O(NlogN) but the communication overhead of the distributed 3-D FFT's make them scale poorly with system size – several all-to-all communication patterns. If we ignore the cost of FFTs (which typically only require >50% of the force computation time for large systems), classical MD simulations scale as O(N) in both memory and CPU cost, where N is the number of particles simulated.

**Other:** Cellulosic Ethanol: A Simulation Model of Lignocellulosic Biomass Deconstruction. Load balancing uses neutral territory methods with staggered domains.

## Biophysics: LAMMPS

*Large-scale Atomic/Molecular Massively Parallel Simulator*

**http://lammps.sandia.gov/**

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.

| | | | | |
|---|---|---|---|---|
| **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<br>fftw<br>fftw_destroy_plan | Creates/destroys an object with information required to compute FFT in the FFTW library | |
| **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:** | A representative 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. Typically, for biomolecular systems, the force computation is dominated by short-range pairwise interactions and long-range Coulomb interactions. The traditional method for solving the long-range Coulomb part are Ewald summations with the solution to the smooth summation part accomplished via distributed 3-D FFTs on a grid to which particle charge density is interpolated. The 3-D FFT’s have a scale computationally O(NlogN) but the communication overhead of the distributed 3-D FFT’s make them scale poorly with system size—several all-to-all communication patterns. If we ignore the cost of FFTs (which typically only require >50% of the force computation time for large systems), classical MD simulations scale as O(N) in both memory and CPU cost, where N is the number of particles simulated. | | | |
| **Other:** | There is no runtime load balancing in LAMMPS. | | | |

## Biophysics: NAMD

*Molecular dynamics code designed for simulation of large biomolecular systems*

**http://www.ks.uiuc.edu/Research/namd/**

NAMD is a parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems. NAMD scales to hundreds of processors on high-end parallel platforms, as well as tens of processors on low-cost commodity clusters, and also runs on individual desktop and laptop computers. NAMD works with AMBER and CHARMM potential functions, parameters, and file formats. NAMD uses the classical molecular dynamics force field, equations of motion, and integration methods along with the efficient electrostatics evaluation algorithms employed and temperature and pressure controls used. Features for steering the simulation across barriers and for calculating both alchemical and conformational free-energy differences are present.

| | | | | |
|---|---|---|---|---|
| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
| | C++ | MPI | Charm++ | None |

| | | | |
|---|---|---|---|
| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** |
| | FFTW | fftw_create_plan<br>fftw<br>fftw_destroy_plan | Creates/destroys an object with information required to compute FFT in the FFTW library |

**Algorithms:** Verlet algorithm to propagate ODEs

**Scaling:** A representative breakdown of CPU cost for a time step is 85% for force computation, 10% for neighbor finding, and 5% includes time integration, application of boundary conditions, etc. Typically, for biomolecular systems, the force computation is dominated by short-range pairwise interactions and long-range Coulomb interactions. The traditional method for solving the long-range Coulomb part are Ewald summations with the solution to the smooth summation part accomplished via distributed 3-D FFTs on a grid to which particle charge density is interpolated. The 3-D FFT's have a scale computationally O(NlogN) but the communication overhead of the distributed 3-D FFT's make them scale poorly with system size – several all-to-all communication patterns. If we ignore the cost of FFTs (which typically only require >50% of the force computation time for large systems), classical MD simulations scale as O(N) in both memory and CPU cost, where N is the number of particles simulated.

NAMD implements runtime load balancing by the use CHARMM++ to migrate work between the processors.

**Other:** 2002 Recipient of the Gordon Bell Award

## *D.2 Chemistry*

### Chemistry: MADNESS

*Multiresolution Adaptive Numerical Scientific Simulation*

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

MADNESS provides a high-level environment for the solution of integral and differential equations in many dimensions using adaptive, fast methods with guaranteed precision based on fast methods and multiwavelet analysis and novel separated representations. There are three main components to MADNESS. At the lowest level is a new petascale parallel programming environment that increases programmer productivity and code performance/scalability while maintaining backward compatibility with current programming tools such as MPI and Global Arrays. The numerical capabilities built upon the parallel tools provide a high-level environment for composing and solving numerical problems in many (1–6+) dimensions. Finally, built upon the numerical tools are new applications with initial focus upon chemistry, atomic and molecular physics, material science, and nuclear structure.

| | | | | |
|---|---|---|---|---|
| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
| | C/C++, Python, Fortran | MPI, Global Arrays | | |
| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** | |
| | BLAS | | | |
| **Algorithms:** | Fast methods with guaranteed precision based on multiwavelet analysis, separated representations of functions and operators, partitioned singular value representations, and bandwidth-limited bases for efficient sampling in space and evolution in time. | | | |
| **Scaling:** | Scaled to 130K+ cores on Jaguar XT5 | | | |
| **Other:** | MADNESS enables scientific applications by addressing the difficulty of solving equations in multi-scale systems implicit from quantum-scale models to simulations of turbulent, reactive flow. Current utilization of MADNESS is in a density functional theory (DFT) application for chemistry with developments in fluid dynamics and climate modeling. | | | |

## Chemistry: NWChem

*Quantum chemistry application*

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

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.

| **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 |

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

**Scaling:** NWChem is made of various modules whose scalability can vary greatly. For example, the DFT module scales between O(N) and $O(N^3)$ (where N is the number of basis functions), while the CCSD(T) codes scales as $O(N^7)$.

**Other:** NWChem is an open-source computational chemistry package for high-performance computing as well as conventional workstation clusters. The science enabled by this application can be seen in the long list of associated publications.

## Chemistry: CP2K

*Atomistic and molecular simulations of solid state, liquid, molecular, and biological systems*

**http://cp2k.berlios.de/**

CP2K is a suite of modules, collecting a variety of molecular simulation methods at different levels of accuracy, from ab-initio DFT to classical Hamiltonians, passing through semi-empirical NDDO approximation. It is used routinely for predicting energies, molecular structures, vibrational frequencies of molecular systems, and reaction mechanisms, and is ideally suited for performing molecular dynamics studies. CP2K provides sophisticated interaction potentials to understand complex reactions at interfaces.

| | | | | |
|---|---|---|---|---|
| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
| | GNU make, Fortran90, CUDA | MPI, OpenMP | | |
| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** | |
| | BLAS/LAPACK, ScaLAPACK, FFTW | pdpotrf, pdsygst, pdsyevd, pdtrsm | | |
| **Algorithms:** | Hamiltonians: Classical, semi-empirical, local and non-local DFT and QM/MM. Algorithms: Molecular Dynamics, Monte Carlo. Free Energy tools and Ehrenfest MD. FFT and sparse linear algebra based. | | | |
| **Scaling:** | Varies from O(N) to O($N^3$) where N is the size of the basis set. | | | |
| **Other:** | CP2K enabled a comprehensive first principles study of the free energy of transfer of hydronium from bulk to interface. | | | |

## Physical Chemistry: QMCPACK

*Quantum Monte Carlo Package*

**http://code.google.com/p/qmcpack/**

The quantum Monte Carlo package was developed at the University of Illinois and is openly released under UIUC/NCSA Open Source License. It is a chemistry code designed for high-performance computers. Simulations start from electronic structure calculations using density functional theory (DFT), Hartree-Fock (HF), and other many-body methods.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | Cmake, C++ | MPI, OpenMP | HDF5, libxml2 | |
| **Math Libraries:** | BLAS/LAPACK, FFTW, einspline, boost | | | |
| **Algorithms:** | Quantum Monte Carlo (QMC) algorithms: Diffusion Monte Carlo (DMC), Variational Monte Carlo (VMC). Particle-based algorithms using dense linear algebra and spline grids for wavefunctions. | | | |
| **Scaling:** | QMC scales as O($N^2$–$N^4$) where N is the number of particles. | | | |
| **Other:** | Quantum Monte Carlo calculation of the energetic, thermodynamics, and structure of water and ice. | | | |

## *D.3 Earth Science*

### Climate: CAM

*Community Atmosphere Model: Numerical modeling of the earth's climate*

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

The general circulation of the atmosphere is modeled by approximations to the primitive equations of geophysical flows in a hydrostatic formulation. These are conservation laws for mass, momentum, energy, and transported constituent species expressed as partial differential and integral equations. A fully active land surface model (CLM) with vegetation modeling and soil hydrology and river routing is included in all CAM simulations.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | Fortran 90 | MPI | NetCDF | |

| | |
|---|---|
| **Math Libraries:** | Cray SciLib,  SGI SCSL |
| **Algorithms:** | CAM can be configured to utilize either of three dynamical cores. The Spectral Eulerian core employs spherical harmonic basis functions to predict the evolution of the large-scale flow. The Semi-Lagrangian Spectral core is run on the same grid, but is formulated in a way that preserves positive-definite behavior for critical advected constituents such as water vapor. The Finite Volume dynamical core employs a finite-difference method. Like the Semi-Lagrangian Spectral core, it also can be configured to guarantee positive-definite behavior for critical advected species. Numerous subgrid-scale physical processes are parameterized, with their effects providing a forcing term to the dynamics. Examples include shortwave and longwave radiative transfer, convective adjustment and clouds. |
| **Scaling:** | The finite volume dynamical core has the best scaling characteristics of the three available options because it is formulated with a 2-D data decomposition (X and Y). The Eulerian and semi-Lagrangian spectral cores only decompose along the Y dimension. Current development will enable scaling to thousands of processors by increasing resolution, adding computational complexity, and implementing more-scalable data distributions. The model is formulated in a hybrid MPI/OpenMP fashion to take advantage of modern cluster architectures. |
| **Other:** | Good connectivity to the Earth System Grid is required. |

## Geoscience: PFLOTRAN

*Modeling reactive flows in porous media*

**http://ees.lanl.gov/pflotran**

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.

| **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. |

**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.

## Ocean Modeling: POP

*Parallel Ocean Program to model ocean circulation in three dimensions*

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

POP is an ocean circulation model derived from earlier models in which depth is used as the vertical coordinate. The model solves the 3-D 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.

| | | | | |
|---|---|---|---|---|
| **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 | | | |
| **Algorithms:** | Spatial derivatives in POP 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. | | | |
| **Scaling:** | Strong scaling to 10K cores. | | | |
| **Other:** | POP is the ocean component of the Community Climate System Model (CCSM) and has been used extensively at LANL in ocean-only mode for eddy-resolving simulations of the global ocean and for ocean-ice coupled simulations with the CICE model. | | | |

## D.4 Engineering

## Combustion: S3D

*Combustion modeling in flames*

**Chen, J. H. et al., "High fidelity simulations for clean and efficient combustion of alternative fuels,"** ***SciDAC2008: Scientific Discovery through Advanced Scientific Computing*** **125:12028 (2008).**

S3D solves a fully coupled system of time-varying partial differential equations (PDEs) governing the full compressible reacting Navier-Stokes, total energy, species 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. In this formulation, 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.

| | | | | |
|---|---|---|---|---|
| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
| | Fortran 90 | MPI | None | Mkdir |

| | | | |
|---|---|---|---|
| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** |
| | None | | |

**Algorithms:** S3D is based on a high-order accurate, nondissipative numerical scheme. It has been used extensively to investigate first-of-a-kind fundamental turbulence–chemistry interactions in combustion topics, including premixed and 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 basically be described as explicit nearest-neighbor local communication. With this design, the code is compute-bound, which has been empirically observed. The code exhibits good weak scaling behavior.

**Other:** Capable of Direct Numerical Simulation of Diesel Jet Flame Stabilization at High Pressure

# *D.5 Fusion*

## Fusion Energy: AORSA

*The All-ORders Spectral Algorithm code: High-resolution solutions for mode conversion and high harmonic fast wave heating in tokamak plasmas*

**http://www.csm.ornl.gov/~shelton/fusion.html**

AORSA 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 $\omega$ 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.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | Fortran 77/90 | BLACS | NetCDF | None |
| **Math Libraries:** | HPL, ScaLAPACK, PBLAS, FFTPACK, PGPLOT | | | |
| **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. 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 (350 × 350) requires the solution of a double complex valued linear system of order 254,823. The larger ITER problem (580 × 580) required to resolve the mode-converted waves requires solution of a linear system of order 68,758. | | | |
| **Scaling:** | Linear scaling up to 48,000 processors; prefer 2–3 times the memory of Jaguar's processors (1.3 Gbytes/processor available to code); domain decomposition with MPI; 50% of peak on Jaguar | | | |
| **Other:** | AORSA could do a complete simulation of mode conversion heating in ITER with a realistic antenna geometry and non-Maxwellian alpha particles. | | | |

## Fusion Energy: GTC

*Gyrokinetic particle simulation of transport barrier dynamics in fusion plasmas*

**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).**

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 the 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 ratio of 2.5%. Using ITG simulations with GTC, they were able to show turbulence spreading for shaped and circle plasmas.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | F90, C | MPI, OpenMP | ADIOS | Timers (through MPI) |
| **Math Libraries:** | PETSc | | | |
| **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 specialized fast Poisson solvers. | | | |
| **Scaling:** | Mature PIC code, nearest-neighbor, good scaling to 5,000 processors, and has been demonstrated on a number of systems utilizing MPI and OpenMPI. 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. GTC has scaled to over 100K cores. | | | |

## Fusion Energy: TGYRO

*Nonlinear tokamak microturbulence package*

**http://fusion.gat.com/theory/Tgyrooverview**

Temperature and density profiles in tokamaks are fundamentally limited by pressure-gradient-driven turbulence and to a lesser extent by cross-field transport caused by collisions so-called neoclassical transport. A first-principles description of these transport processes can be obtained via direct kinetic simulations, but to date, these have been far too computationally expensive to be considered for modeling and performance-prediction purposes. To solve the inverse problem, modelers use reduced models for core thermal and particle transport where a typical modeling scenario may require the evaluation of thousands of local transport fluxes. By developing an iteration scheme suitable for solving the inverse problem, we have been able to obtain steady-state temperature profiles for DIII-D plasmas using GYRO to repeatedly calculate turbulent particle and energy fluxes. These so-called transport solutions also include first principles neoclassical fluxes computed using NEO. TGYRO oversees execution of multiple simultaneous instances of both GYRO and NEO.

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.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | Fortran 77/90 | MPI | MPIIO | Timing only |
| **Math Libraries:** | BLAS/LAPACK, UMFPACK, MUMPS, FFTW | | | |
| **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 gyro averages 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 through 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). | | | |

# *D.6 Materials*

## Materials Science: DCA++

*Quantum Monte Carlo calculations for materials characterization*

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

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). DCA++ is based on a dynamic cluster quantum Monte Carlo algorithm to solve, in a controlled way, lattice models of strongly correlated electron systems such as the 2-D Hubbard model. 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.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | C++ | MPI | None | None |

| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** |
|---|---|---|---|
| | BLAS | | |

**Algorithms:** 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 code to be run for large problems with high efficiency on superscalar processors.

**Other:** 2008 Recipient of the Gordon Bell Award

## Materials Science: gWL-LSMS

*Electronic structure calculations based on density functional theory*

**http://www.ccs.ornl.gov/mri/repository/LSMS**

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 won the Gordon Bell prize in 2009.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | Fortran 77/90, C++ | MPI2 | HDF5 | 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 |

**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 inversion of 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 depend only 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:** Parallelization is achieved by assigning system atoms to different processors. 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 >200,000 cores when investigating the thermodynamic behavior of 1,000–10,000 atom nanoparticles.

**Other:** 2009 Recipient of the Gordon Bell Award

## Materials Science: VASP

*Vienna Ab-initio Simulation Package for molecular dynamics simulations of large biomolecular systems*

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

Plane wave-based density functional calculations, together with all-electron-derived pseudo potentials, comprise 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.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | Fortran90, C (minor) | MPI | None | Getrusage |

| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** |
|---|---|---|---|
| | *BLAS* | *D/ZGEMM* | Double complex/real general matrix-matrix multiply |
| | *BLAS* | *(D/Z)TRMM* | Double complex/real triangular matrix- matrix multiply |
| | | *P(D/Z)TRTR* | |
| | *ScaLAPACK* | *P(D/Z)POTRF* | Matrix inverse, Cholesky decomposition, Eigenvector computation |
| | | *P(D/Z)HEEVX* | |

**Algorithms:** Planewave code that solves 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 offers much improved accuracy and reduced computational costs (flops and memory) over the simpler norm-conserving potentials, particularly for systems containing transition metal atoms. 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.

**Scaling:** Although the method is in-principle cubic scaling, in practice it scales quadratically up to 1000 atoms using recent numerical advances. Appropriately configured, VASP currently delivers a large fraction of peak performance, typically 30–50%, for up to 1000 processors.

## Nanoelectronics: OMEN

*Atomistic and full-band quantum transport simulator designed for post-CMOS devices*

**http://cobweb.ecn.purdue.edu/~gekco/omen**

OMEN is a two- and three-dimensional Schrodinger-Poisson solver based on the $sp^3d^5s^*$ semi-empirical tight-binding method. This bandstructure model has been chosen for (i) its ability to reproduce the principal bulk characteristics of electrons and holes, (ii) its straightforward extension to nanostructures, and (iii) its atomic description of the simulation domain. Carrier and current densities are obtained by injecting electrons and holes at different energies into the device and by solving the resulting system of equations in the Wave Function or in the Non-equilibrium Green's Function formalism. The 2-D or 3-D Poisson equation is expressed in a finite-element basis where the tight-binding charges sit on node position. OMEN will enable the calculation of full transistor characteristics of realistic semiconductor devices, using quantum mechanics in an atomistic representation.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | C/C++, Fortran | | | |

| | |
|---|---|
| **Math Libraries:** | SuperLUdist, MUMPS, PARPACK, LAPACK/BLAS |
| **Algorithms:** | Recursive Green's Function; The Schrödinger equation is solved with open boundary conditions (OBCs) either in the Wave Function (WF) or in the Non-Equilibrium Green's Function (NEGF) formalism. The resulting charge and current densities are self-consistently coupled to the Poisson equation in a finite-element basis. |
| **Scaling:** | Scaled to 222,730 cores. The computation of the bias points, the energy and momentum integrations, as well as the spatial domain decomposition have been parallelized so that a single simulation can run on a number of processors Ncpu up to $O(10^4)$ with a speed-up factor close to Ncpu. |
| **Other:** | OMEN enables discovery of new nanoscale technologies for faster switching, smaller feature size, and reduced heat generation. The creation of a new switch will revitalize the semiconductor industry in 2015. Designers will be enabled to directly address questions of quantization and spin, tunneling, phonon interactions, and heat generation for nanoscale devices. |

## Nanoscience: LS3DF

*Linear Scaling 3-D Fragment code for electronic structure calculations*

**Lin-Wang Wang, "Linear Scaling 3-D Fragment Method for Large-Scale Electronic Structure Calculations," Lawrence Berkeley National Laboratory (2008).**

New linearly scaling three-dimensional fragment (LS3DF) method for large scale ab initio electronic structure calculations. LS3DF is based on a divide-and-conquer approach, which incorporates a novel patching scheme that effectively cancels out the artificial boundary effects due to the subdivision of the system. As a consequence, the LS3DF program yields essentially the same results as direct density functional theory (DFT) calculations at a much smaller computational cost.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | Fortran | MPI | | |

| | |
|---|---|
| **Math Libraries:** | ESSL, BLAS2, LAPACK, FFT |
| **Algorithms:** | LS3DF divides a large system into small pieces called fragments, and then calculates the electron wavefunctions and the charge density of each piece independently using a small group of processors. The charge densities of all the pieces are then patched together to determine the charge density of the entire system. Finally, a Poisson equation is solved for the whole system charge density, until a self-consistent charge density for the entire system is achieved. |
| **Scaling:** | The fragments of the LS3DF algorithm can be calculated separately with different groups of processors. This leads to almost perfect parallelization on tens of thousands of processors. After code optimization, they were able to achieve 35.1 Tflop/s, which is 39 percent of the theoretical speed on 17,280 Cray XT4 processor cores. Their 13,824-atom ZnTeO alloy calculation runs 400 times faster than a direct DFT calculation, even presuming that the direct DFT calculation can scale well up to 17,280 processor cores. These results demonstrate the applicability of the LS3DF method to material simulations, the advantage of using linearly scaling algorithms over conventional $O(N^3)$ methods, and the potential for petascale computation using the LS3DF method. |
| **Other:** | LS3DF simulations allow for a better understanding of nanostructure solar cells to improve their efficiency and viability as a mainstream solution for renewable solar energy. |

## *D.7 Nuclear Energy*

### Neutron Transport: Denovo

*Nuclear reactor neutron transport solver*

Progress in nuclear technology requires simulations that model many coupled physics systems, including Boltzmann transport equations, a powerful algorithm for analyzing transport phenomena with many-step gradients in both density and temperature. For nuclear reactor simulation, the size of the equations to be modeled is tremendous—five orders of magnitude in space and ten in neutron energy. Denovo is a parallel transport solver which is the first-of-a-kind, mathematically consistent, two-level approach to the multiscale challenge. With present algorithms, a solver that incorporates the separating out of processes for all scales would require $10^{17}$ to $10^{21}$ degrees of freedom (DOF) for a single time-step, which is beyond even exascale computational resources. Denovo is a significant advance over current technology, because it allows fully consistent multi-step approaches to high-fidelity nuclear reactor simulations that cannot be performed with current technology.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | C++, Python, F95 | MPI | Silo, HDF5 | |

| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** |
|---|---|---|---|
| | Trilinos | | |

**Algorithms:** Denovo incorporates a wavefront algorithm to sweep across spatial domain. The angular domain is in the process of being thread-parallelized and the energy domain will be decomposed as well. We are evaluating the performance of several acceleration/solver approaches to the eiganvalue form of the transport equation, including the Trilinos/Anasazi solvers, a traditional non-linear CMFD approach, and a shifted-eigenvalue coupled with Krylov solver.

**Scaling:** Denovo scales to the petaflop level by uncoupling the multi-level, phase-space parameters of the equations.

**Other:** This multi-level approach with advanced computing can provide an extremely high-fidelity capability to understand the power distribution within a nuclear reactor in an approach that can be coupled with computational fluid dynamics and conjugate heat transfer solvers to understand the performance of a nuclear reactor during nominal, and potentially transient, operating conditions.

## Neutron Transport: UNIC

*Neutron transports simulations in nuclear reactors*

UNIC is developed at ANL as part of the DOE Nuclear Energy Advanced Modeling and Simulation (NEAMS) program. UNIC models how neutrons move inside the reactor core. The simulation of processes in reactor cores is challenging because of large length scales, a complicated distribution of materials, and the intricacies of the physical data. Calculating and simulating these processes requires simulation over several orders of magnitude and energy, and the resolution of strong local fluctuations. UNIC solves large-scale nuclear reactor core problems governed by the seven dimensional (three in space, two in angle, one in energy, and one in time) Boltzmann equation. The goal of this simulation effort is to reduce the uncertainties and biases in reactor design calculations by progressively replacing existing multi-level averaging (homogenization) techniques with more direct solution methods. As the algorithms are refined, they will be used to solve coupled physics problems in such a way that thermal, hydraulic, and structural feedbacks are accurately represented in realistic reactor transient simulations. This will lead to a significant reduction in cost and better assessments of the safety of fast reactors—nuclear reactors where the fission is sustained by fast neutrons.

| | | | | |
|---|---|---|---|---|
| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
| | C++ | MPI | | |
| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** | |
| | PETSc, MeTIS | | | |
| **Algorithms:** | Currently UNIC has two solvers for the neutron transport equation which are based upon the second-order even parity transport equation and utilize a spherical harmonics and a discrete ordinates approximation for the angular approximation. A third solver based on a first-order method of characteristics has also been implemented in order to provide a more efficient capability of explicit geometry modeling. UNIC uses an unstructured mesh and to represent the complex geometry of a reactor core, billions of spatial elements, hundreds of angles, and thousands of energy groups are necessary, which leads to problem sizes with petascale degrees of freedom. | | | |
| **Scaling:** | UNIC shows weak scalability of over 80% on 131,072 cores on Jaguar XT5. | | | |
| **Other:** | Scalable simulation of unstructured, deterministic neutron transport in fast reactor cores. | | | |

## *D.8 Physics*

### Astrophysics: CHIMERA

*3-D modeling of core collapse supernovae*

**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).**

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. In addition, thermonuclear kinetics are evolved in each spatial zone via a local nuclear burning module.

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | F90 | MPI | HDF5, pnetCDF | None |

| | |
|---|---|
| **Math Libraries:** | LAPACK |
| **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. 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. |
| **Other:** | The world's first core-collapse supernova simulations in 3-D with realistic neutrino transport. The simulations would also likely include magnetic fields, some approximation to general relativistic gravity, and realistic nuclear burning. |

## Astrophysics: FLASH

*Modular, parallel, multiphysics simulation code for handling general compressible flow problems found in many astrophysical environments*

**http://flash.uchicago.edu**

FLASH is designed to solve compressible, reactive flow problems in dense stellar environments, like those found in novae, X-ray bursts, and Type Ia supernovae. 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.

| | | | | |
|---|---|---|---|---|
| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
| | Python, F90, C | MPI | HDF5, pnetCDF | GNU make |
| **Math Libraries:** | No external libraries | | | |
| **Algorithms:** | FLASH uses an explicit, PPM-based method, hence is 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 platform. The code exhibited good scaling. | | | |
| **Other:** | 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. | | | |

## Condensed Matter: CASINO

*First-principles electronic structure calculations using Quantum Monte Carlo methods*

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

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.

| | | | | |
|---|---|---|---|---|
| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
| | Fortran 90 | MPI | None | Timing |
| **Math Libraries:** | BLAS | | | |
| **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 to a failed processor can be compensated for with only minor overhead. | | | |
| **Scaling:** | The computational requirements scale with the second to fourth power of the number of electrons and atoms, depending on the quantities being measured. 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 1,000–10,000 electrons on 10,000–100,000 processors without major developments to existing code. 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. | | | |
| **Other:** | 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. | | | |

## Lattice Gauge Theory: MILC/Chroma

*Numerical studies of quantum chromodynamics*

**MILC: http://physics.indiana.edu/~sg/milc.html**
**CHROMA: http://usqcd.jlab.org/usqcd-docs/chroma/**

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.

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.

| **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) |

| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** |
|---|---|---|---|
| | None | | |

**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:** 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.

## Nuclear Physics: MFDn

*Many Fermion Dynamics – nuclear*

**https://hpcrd.lbl.gov/scidac09/talks/Vary_SciDAC_2009.pdf**

MFDn is a state-of-the-art configuration interaction nuclear shell model application, capable of calculating the energy spectrum, wavefunctions and observables for light nuclei, using two- and three-body interactions. One major challenge for nuclear science is to provide a consistent theoretical framework that accurately describes all nuclei, and can be used predictively. Toward this aim, MFDn allows for ab initio investigations with two-body and three-body interactions derived from effective field theory within the microscopic no-core shell model approach. These interactions provide a long sought after bridge between low-energy nuclear physics and quantum chromodynamics (QCD), the fundamental theory describing the strong interaction.

| | | | | |
|---|---|---|---|---|
| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
| | Fortran90 | MPI, OpenMP (optional) | MPI-IO | |
| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** | |
| | None | | | |
| **Algorithms:** | MFDn solves the nuclear many-body problem as a large-sparse matrix eigenvalue problem. It generates the governing Hamiltonian matrix and uses the Lanczos algorithm (iterative Krylov solver) to solve for the lowest eigenvalues of interest. | | | |
| **Scaling:** | Problem size scales by number of particles and size of model space allowed. Features intense integer arithmetic operations. Currently scales to 200K+ cores on the hex-core upgraded Jaguar XT5. | | | |
| **Other:** | Used to investigate the nuclear structure of carbon-14 to explain its anomalously long half-life useful in carbon-dating. | | | |

## Nuclear Physics: NUCCOR

*Nuclear Coupled-Cluster Oak Ridge*

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

Nuclear scientists strive to understand the properties of atomic nuclei from the interactions of protons and neutrons, or ultimately, from quarks and gluons. Indeed, nuclei are the only link between QCD and the atomic and macroscopic world. For medium mass nuclei, the coupled-cluster approach is the only microscopic method presently available. NUCCOR iteratively solves for the one- and two-body amplitudes of the coupled, non-linear algebraic CCSD (coupled-cluster with single and double excitations) equations that arise from application of an exponentiated cluster excitation operator onto a reference many-body wave-function (usually taken as a single Slater determinant).

| **System Software:** | ***Programming languages:*** | ***Communication libraries:*** | ***I/O libraries and functions:*** | ***Operating system functions:*** |
|---|---|---|---|---|
| | F90 | MPI | MPI-IO | None |

| **Math Libraries:** | ***Library*** | ***Function*** | ***Functionality*** |
|---|---|---|---|
| | BLAS | | Matrix-matrix; matrix-vector |
| | BLAS | | Tensor-tensor multiplies of (size 1002 × 1002 and 10004 (100 particles and 1000 basis states)) |

**Algorithms:** Solves a nonlinear set of coupled algebraic equations. A complete calculation for a given nucleus proceeds in the following manner. 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. 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. 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 with an allreduce to obtain the new copies of the amplitudes for the next iteration step.

**Scaling:** The computational requirements scale as $No^2Nu^4$, where No and Nu are the number of occupied and unoccupied single-particle orbitals, respectively.

**Other:** NUCCOR is the workhorse for the developing time-dependent coupled cluster application that will provide the first ever microscopic calculations of nuclear fission.