Application Requirement Analysis for the LCF

O. E. Bronson Messer, Doug Kothe, & Ricky Kendall
National Center for Computational Sciences
Oak Ridge National Laboratory

Abstract

We have attempted to organize, analyze, and interpret several sets of empirical data regarding LCF applications with an eye towards using actual science requirements to guide future system design. Much of this analysis has centered on the application of many different methodologies to the same sets of data. Perhaps surprisingly, many of these attempts have provided a somewhat consistent picture of what future NLCF platforms must be capable of to satisfy most application requirements.
KEYWORDS: applications, architectures, performance, programming environments

1 Introduction

A requirement is a condition or capability needed by a user to solve a problem or achieve an objective or is a condition or capability that must be met or possessed by a system to satisfy a contract, standard, specification, or other formally imposed document. Both definitions apply for the breakthrough computational science requirements used in the design, procurement, deployment, and operation of the Department of Energy (DOE) Leadership Computing Facility (LCF) at the National Center for Computational Sciences (NCCS). By articulating these requirements and using them to manage and arbitrate decisions, the NCCS will align LCF systems to the maximum extent possible with the needs and goals of the breakthrough science projects using these resources. LCF requirements for the NCCS apply to the entire end-to-end analysis process followed by scientists using the NCCS facilities, from system hardware, to system software, to the integrated development environment, and, finally, to the problem solving environment, which includes data analysis, management, and visualization. We expect that effective requirements development, management, and planning will positively influence the design, procurement, deployment, and operation of an NCCS system by improving the quality, quantity, or fidelity of the output of one or more breakthrough science simulation applications in a measurable way. For requirements to be useful to the NCCS, they must be actionable, i.e. as quantitative as possible, without being solutions. In reality, requirements flow in both directions: applications impose requirements on the LCF systems, and the LCF systems in turn impose real requirements upon the applications.

A valid requirements process must follow three basic steps: development, management, and planning. The NCCS requirements effort in 2006 was principally devoted to establishing the methods by which the three-step requirements process is executed, and the first step in the requirements process, namely requirements development. Key in requirements development is elicitation , which is an ongoing process involving both analysis of existing documentation as well as direct queries of users. Information from users was elicited in three separate instances that were either distilled from previous information from science teams or questionnaires given directly to the science project team members:

- A requirements survey constructed and collected by the NCCS Application Requirements Council (ARC);
- A code project survey constructed by the Advanced Scientific Computing Advisory Committee (ASCAC) sub-panel on science metrics for the Advanced Scientific Computing Research (ASCR) computing facility metrics; and
- Answers to science and code questions embodied in the LCF and INCITE project proposal applications.

For the 22 projects allocated on the NCCS LCF systems in 2006, 8 projects responded to the ARC sur-

vey, 19 responded to the ASCAC survey, and all 22, of course, filled out proposal applications (necessary for allocation awards). Answers to these surveys helped to define requirements from the following points of reference: science motivation and impact, science quality and productivity, application models, application algorithms, application software, application footprint, and data management and analysis. For the most part, our first attempts at synthesizing these data has been of three types: understanding what scientific questions are currently being addressed, what scientific questions are hoped to be addressed as we reach 250 TF and 1 PF of peak computational capability, and what are the machine and infrastructure characteristics most important to examine to reach these scientific goals.

2 Science Drivers

Over the past five years, the DOE Office of Science SciDAC-1 Program has achieved simulation-based scientific accomplishments through focused collaboration and active partnership of domain scientists, applied mathematicians, and computer scientists. The LCF at NCCS has played a role in many of these successes, e.g., nanoscience, accelerator design, astrophysics, chemistry, combustion, climate modeling, and fusion. Even more compelling opportunities for scientific discovery have fostered a new SciDAC-2 Program in which a series of coordinated investments across all DOE/SC Programs (Basic Energy Sciences, Biological and Environment Research, Fusion Energy Sciences, High-Energy Physics, and Nuclear Physics) promises to further the achievement of breakthrough science through simulation through (1) focused efforts on scientific applications in specific domain and (2) enabling technologies in computer science, software infrastructure, and applied mathematics through Centers for Enabling Technologies (CETs), university-led Institutes, and Scientific Application Partnerships (SAPs). SciDAC-2 thrust areas (with examples) include accelerator science (ILC design), astrophysics (understanding of nucleosynthesis), climate modeling (global carbon cycle prediction), biology (protein interaction networks), fusion (ITER design), groundwater (subsurface reactive transport), high energy physics (dark universe and neutrinos), nuclear physics (NNSA physics), and QCD (lattice gauge theory). Other science areas ripe for discovery include nanoscience, chemistry, nuclear energy, and manufacturing. Many successful INCITE proposals are associated with these science teams, and they laid out in their proposals for FY2007 a raft of scientific milestones for their projects. The current state-of-the-art in various computational science domains, and best guesses for increased physical fidelity are given in Figure 1.

3 Algorithms and Implementations

The algorithms used by computational scientists in various fields do much to determine the scaling and performance capabilities of the codes they build and use. For the most part, the choice of algorithm sets an envelope of performance characteristics that is then filled to greater or lesser extent by the implementation of the algorithm on a given computational platform. Therefore, algorithms and implementations should be considered together, and in tandem with the hardware and software environment chosen.

A prerequisite to implementing any algorithm is decided the software stack, including programming languages and libraries, among other things, that will be used to perform the implementation. This choice is unique to individual codes. Current constraints of this type can be seen in Figure 2

Once the software stack is set, one can turn to algorithmic characteristics. A popular foil to describe the algorithmic characteristics of a given application is the so-called "7 dwarfs" first promulgated by P. Colella. (N.B. the "dwarfs" considered here may well be slightly different from the lists of others, as the concept has tended to persist, while the specific algorithmic tropes have tended to change based on the purposes of the author). Shown in Figure 3 is our best understanding of how the applications we interrogated fit in the "7 dwarfs" space.

Several trends are notable in the "7 dwarfs" categorization of codes in Figure 3:

- The 7 algorithm types are scattered broadly among science domains, with no one particular algorithm being ubiquitous and no one algorithm going unused;
- Structured grids and dense linear algebra algorithms are the most widely used algorithms (by over half of the representative codes), hence system attributes such as node peak flops and memory capacity, memory latency, and interconnect latency will be important;

Science Domain	Code	Fidelity @ 25 TF	Fidelity @ >1 PF
Astrophysics	Chimera	3D hydro simulations to follow the shock evolution out to several times the stalled shock radius.	3D Boltzmann transport with 20 x 16 neutrino phase space resolution (20 energy groups x 16 angles in an Sn-type scheme).
Astrophysics	Vulcan2D	2D multi-group, time-dependent radiation hydrodynamics with 10000 km and 2 s resolution.	3D multi-group, time-dependent radiation hydrodynamics. More integration time, more state variables, increased complexity reaction networks.
Biology	LAMMPS	Dynamics of 700K atom systems for 5-10 ns of model time per day of simulation time.	Multi-million atom systems for 0.1-1.0 ms.
Climate	CCSM	Eulerian spectral atmospheric circulation model with diurnal cycle resolved columnar radiation and moist convection (CAM3), Brian-Cox free-surface ocean model with Gent-McWilliams eddy parameterization (POP1.4), dynamic sea-ice model with visco-plastic rheology (CICE), land surface model with soil, river and vegetation components (CLM3).	Tropospheric chemistry (100 species), dynamic vegetation, terrestrial carbon pools, ocean ecosystems, land ice sheets, stratospheric chemistry, full sulfur cycle, increase in ensemble size for climate change studies, coupled-ocean eddy-resolving simulations, cloud microphysics, realistic land-use patterns, tropical event simulation on climate timescales.
Climate	MITgcm	4km horizontal and variable (30 levels) vertical resolution in a 2000 by 4000 km domain 2000m deep. Time step of 500 s and integration of the primitive equations (3 diagnostic and 2 prognostic variables) for several decades (20-40 years).	Understand role of non-hydrostatic physics and internal wave breaking in deep ocean mixing, required to close the heat and salinity budget suggested by the current sinking rates in high latitudes and in the establishment of the thermohaline circulation.
Combustion	S3D	Chemical mechanism for CO/H_2 and reduced mechanism for CH_4 and molecular transport model. 2.5 decades of time and length scales resolved for reactive turbulent flow. Moderate Reynolds numbers of 5-15K.	Increase Reynolds numbers to >20K, (consistent with internal combustion engines) and pressures to 10-20 atm, Chemical mechanisms include multi-stage n-heptane ignition.
Fusion	GTC	Gyrokinetic ions with drift-kinetic electrons and electrostatic perturbations. Resolved time scale is the electron transit time and the resolved length scale is the ion gyroradius.	Integrated simulation with gyrating ions and drift-kinetic electrons with electromagnetic perturbations by resolving ion cyclotron waves and electron skin depth. Transport time scale simulations (evolving plasma background equilibrium).
Fusion	GYRO	Testing first-principles models of plasma turbulence against measured levels of heat and particle transport in tokamaks (DIII-D, C-mod, NSTX) to build reduced models of plasma transport to predict performance of prototypes.	ITER performance predictions. Possible to introduce feedback loop to adjust input profiles to match target heat and particle flows for truly predictive simulation.

Figure 1: Increase in science simulation fidelity possible with a 1 PF LC system for specific application codes in various science domains.

Science Domain	Code	Programming Language	Programming Model	I/O Libraries	Math Libraries
Accelerator Design	T3P	C/C++	MPI	NetCDF	MUMPS, ParMETIS, Zoltan
Astrophysics	CHIMERA	F90	MPI	HDF5, pNetCDF	LAPACK
Astrophysics	VULCAN/2D	F90	MPI	HDF5	PETSc
Biology	LAMMPS	C/C++	MPI		FFTW
Chemistry	MADNESS	F90	MPI		BLAS
Chemistry	NWChem	F77, C/C++	MPI, Global Arrays, ARMCI		BLAS, ScaLAPACK, FFTPACK
Chemistry	QReTran	F95	MPI		LAPACK
Climate	CAM	F90, C, CAF	MPI, OpenMP	NetCDF	SciLib
Climate	POP/CICE	F90, CAF	MPI, OpenMP	NetCDF	
Climate	MITgcm	F90, C	MPI, OpenMP	NetCDF	
Combustion	S3D	F90	MPI		
Fusion	AORSA	F77, F90		NetCDF	ScaLAPACK, FFTPACK
Fusion	GTC	F90, C/C++	MPI, OpenMP	MPI-IO, HDF5, NetCDF, XML	PetSC
Fusion	GYRO	F90, Python	MPI	MPI-IO, NetCDF	BLAS, LAPACK, UMFPACK, MUMPS, FFTW, SciLib, ESSL
Geophysics	PFLOTRAN	F90	MPI		BLAS, PetSC
Materials Science	LSMS	F77, F90, C/C++	MPI2	HDF5, XML	BLAS, LAPACK
Materials Science	QBOX	C/C++	MPI	XML	LAPACK, ScaLAPACK, FFTW
Materials Science	QMC	F90	MPI		BLAS, LAPACK, SPRNG
Nanoscience	CASINO	F90	MPI		BLAS
Nanoscience	VASP	F90	MPI		BLAS, ScaLAPACK
Nuclear Energy	NEWTRNX	F90, C/C++, Python		HDF5	LAPACK, PARPACK
Nuclear Physics	CCSD	F90	MPI	MPI-IO	BLAS
QCD	MILC, Chroma	C/C++	MPI		

Figure 2: Functional software requirements for specific application codes in various science domains.

Science Domain	Code	Structured Grids	Unstructured Grids	FFT	Dense Linear Algebra	Sparse Linear Algebra	Particles	Monte Carlo
Accelerator Physics	T3P		X			X		
Astrophysics	CHIMERA	X			X	X	X	
Astrophysics	VULCAN/2D		X		X			
Biology	LAMMPS			X			X	
Chemistry	MADNESS		X		X			
Chemistry	NWCHEM			X	X			
Chemistry	OReTran	X		X	X			
Climate	CAM	X		X			X	
Climate	POP/CICE	X				X	X	
Climate	MITgcm	X				X	X	
Combustion	S3D	X						
Fusion	AORSA	X		X	X			
Fusion	GTC	X				X	X	X
Fusion	GYRO	X		X	X	X		
Geophysics	PFLOTRAN	X	X			X		
Materials Science	QMC/DCA				X			X
Materials Science	QBOX			X	X		X	
Nanoscience	CASINO						X	X
Nanoscience	LSMS	X			X			
Nuclear Energy	NEWTRNX		X		X	X		
Nuclear Physics	CCSD				X			
QCD	MILC	X						X

Figure 3: Colellas "7 dwarfs" categorization of algorithms employed by specific application codes in various science domains. An X entry denotes a particular algorithm that is utilized by that code.

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

These algorithm characteristics and their associated implementations lead naturally to a set of application hardware requirements. A given algorithmic tends to stress different parts of a computational platform. Examples of these kinds of requirements are shown in Figure 2.

Given these requirements, and the understanding of individual scientists regarding what they need from computation, one can attempt to rank components of a LCF as regards their relative importance for a given application. This is, perhaps, a dangerous exercise, as the tendency might be to regard some components as "important" and others as "not important." This is to be avoided, as a balanced system seems to be the only reasonable response. Nevertheless, an example of such ranking can be seen in Figure 5.

4 A Few Conclusions

The end result of the described data collection and analysis is not yet a coherent, whole set of clear recommendations. However, some impressions and, at least anecdotally supported, claims can be stated.

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

- Parallel algorithm maturity and efficiency vary widely from one field to another and from one code to another. For example, fields focused on "atomistics" (nanoscience, materials science, chemistry, biology, etc.) have parallelism challenges that are unique enough to make it difficult for other fields to contribute useful approaches.

- A "7 dwarfs" algorithm analysis of applications indicates, not surprisingly, that there are no algorithm sweet spots, thereby disallowing an LCF system to pursue a hardware architecture designed to specifically optimize a particular algorithm (e.g., FFT).

- Standard programming languages, e.g., Fortran, C, and C++, remain the scientific computing staple on LCF systems. To a lesser extent, Co-array Fortran and scripting languages like Python are also needed, but a demand for brand new and/or unanticipated languages is not evident at present.

- Parallel programming strategies continue to emphasize MPI, along with, in some cases, OpenMP and Global Arrays. Other paradigms need to be examined, at least in prototype form, in order to demonstrate proof of principle.

- "Critical" math libraries needed by a large fraction of applications include BLAS, LAPACK, FFTPACK, FFTW, and PETSc. Others needed (but not as prevalent) include ParMETIS, MUMPS, and Zoltan.

- Most applications have chosen to implement fault-tolerance via their own checkpoint restart capability rather than rely on the need for a fault-tolerant communication library. Further possibilities in this regard should be pursued.

- Creative hybrid parallel programming models for efficient scaling on multi-core processors need to be pursued vigorously. And this is not necessarily just a clever combination of MPI and OpenMP.

- Large-scale application codes can easily have useful lifetimes of 20-50 years (corresponding to 5-10 generations of LCF systems), with the first 5-10 years (and ≈100 man-years of effort) used just to reach maturity. Expecting applications code developers to rewrite a mature code from scratch (e.g., in a new language, etc.) in order to achieve better scaling or parallel performance is therefore nave. Applications code developers are talented; they are

LC System Attribute	Application Algorithms Driving a Need for this Attribute	Application Behaviors Driving a Need for this Attribute
Node Peak Flops	Dense Linear Algebra, FFT, Sparse Linear Algebra, Monte Carlo	Scalable and required spatial resolution low; would benefit from a doubling of clock speed; only a problem domain that has strong scaling, completely unscalable algorithms; embarrassingly parallel algorithms (e.g., SETI at home)
Mean Time to Interrupt	Particles, Monte Carlo	Naïve restart capability; large restart files; large restart R/W time
WAN Bandwidth		Community data/repositories; remote visualization and analysis; data analytics
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 (PIC)
Local Storage Capacity	Particles	High frequency/large dumps, out-of-core algorithms, debugging at scale
Archival Storage Capacity		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, etc.); expensive to recreate; nowhere to store elsewhere
Memory Latency	Sparse Linear Algebra	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		Naïve out-of-core memory usage; many small I/O files; small record direct access files;
Interconnect Bandwidth	Dense Linear Algebra (global), Sparse Linear Algebra (global), Unstructured Grids	Big messages, global reductions of large data; implicit algorithm with large DOFs per grid point;
Memory Bandwidth	Sparse Linear Algebra, Unstructured Grids	Large multi-dimensional 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		Reads/writes large amounts of data at a relatively low frequency; read/writes lots of large intermediate temporary data; well-structured out-of-core memory usage

Figure 4: Science application behavioral and algorithmic drivers for LC system attributes.

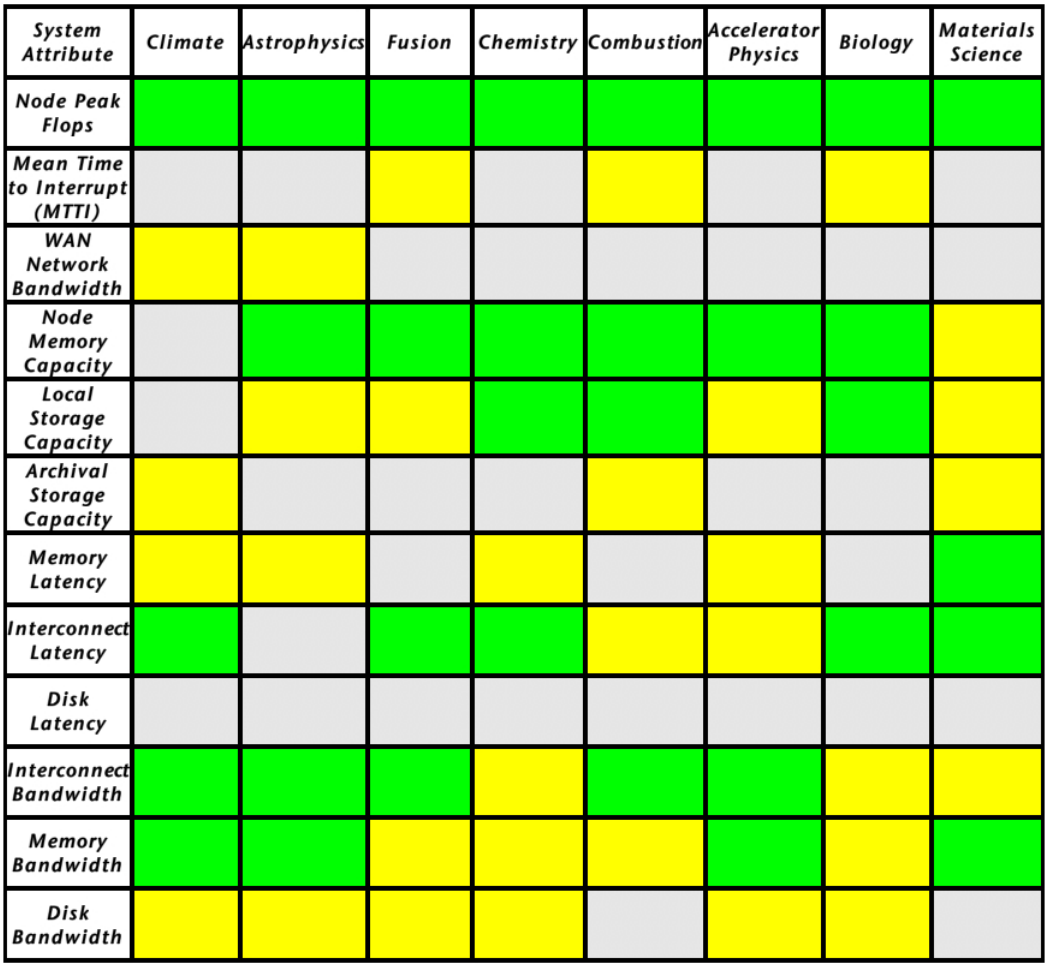

Figure 5: Three-tier prioritization of twelve system attributes for relevant science domains. In each science domain, green attributes as those with the highest priority for maximizing, yellow is moderate priority, and grey lowest priority.

adept at and used to refactoring existing code to achieve better performance. This will be the approach of preference on petascale LCF systems. There is no magic language or compiler that can do better in this short time frame.

- With petascale LCF systems consisting of 100K (or more) nodes and/or processors, parallel algorithms must not only work, but their implementation in software must possess the highest SQA standards. Software quality, and the breadth and depth of testing required to ensure and maintain this quality, is too often underemphasized or neglected under the pressure of producing timely science results. This trend could be exacerbated on LCF systems.
- The path forward for many application areas includes either enhanced resolution or additional physics or both. This necessarily translates to increased aggregate and per-node memory requirements. Given the present cost of memory relative to processing power, this requirement represents a fundamental tension that must be carefully examined.
- Developer estimates for many code characteristicse.g. memory usage, network bandwidth, wallclock timeare often completely swamped by poor implementations of algorithms and other software infrastructure. A basic understanding of fundamental algorithm characteristicse.g. floating point operations required, memory operations requiredis necessary to accurately evaluate such requirements.

5 About the Authors

Bronson Messer is an R&D Staff Member in the Scientific Computing Group of the National Center for Computational Sciences at Oak Ridge National Laboratory. He spends some of his time thinking about application performance and requirements. He can be reached at bronson-at-ornl-dot-gov.

Doug Kothe is the Director of Science at the NCCS. He can be reached at kothe-at-ornl-gov.

Ricky Kendall is the group leader for Scientific Computing at the NCCS. He can be reached at kendallra-at-ornl-dot-gov.