LBNL-1014E

NERSC-6 Workload Analysis and Benchmark Selection Process

Katie Antypas, John Shalf, and Harvey Wasserman

National Energy Research Scientific Computing Center Division
Ernest Orlando Lawrence Berkeley National Laboratory
Berkeley, CA 94720

August 13, 2008

This work was supported by the U.S. Department of Energy's Office of Science, Office of Advanced Scientific Computing Research, Mathematical, Information and Computational Sciences Division under Contract No. DE-AC02-05CH11231.

Abstract

This report describes efforts carried out during early 2008 to determine some of the science drivers for the "NERSC-6" next-generation high-performance computing system acquisition. Although the starting point was existing Greenbooks from DOE and the NERSC User Group, the main contribution of this work is an analysis of the current NERSC computational workload combined with requirements information elicited from key users and other scientists about expected needs in the 2009-2011 timeframe. The NERSC workload is described in terms of science areas, computer codes supporting research within those areas, and description of key algorithms that comprise the codes. This work was carried out in large part to help select a small set of benchmark programs that accurately capture the science and algorithmic characteristics of the workload. The report concludes with a description of the codes selected and some preliminary performance data for them on several important systems.

NERSC-6 Workload Analysis and Benchmark Selection Process

Introduction

NERSC's science-driven strategy for increasing research productivity focuses on the balanced and timely introduction of the best new technologies to benefit the broadest subset of the NERSC workload. The goal in procuring new systems is to enable new science discoveries via computations of a scale that greatly exceed what is possible on current systems.

The key to understanding the requirements for a new system is to translate scientific needs expressed by simulation scientists into a set of computational demands and from these into a set of hardware and software attributes required to support them. This combination of requirements and demands is abstracted into a set of representative application benchmarks that by necessity also capture characteristic styles of coding and thus assure a workload-driven evaluation. The importance of workload-based performance analysis was best expressed by Ingrid Bucher and Joanne Martin in a 1982 LANL technical report where they stated, *"Benchmarks are only useful insofar as they model the intended computational workload."*

At NERSC these carefully chosen benchmarks go beyond representing a set of individual scientific and computational needs. The benchmarks also comprise the Sustained System Performance (SSP) metric, an aggregate measure of the real, delivered performance of a computing system. The SSP is evaluated at discrete points in time and also as an integrated value to give the system's potency, meaning an estimate of how well the system will perform the expected work over some time period. Taken together, the NERSC SSP benchmarks, their derived aggregate measures, and the entire NERSC workload-driven evaluation methodology create a strong connection between science requirements, how the machines are used, and the tests we use.

This document begins with an examination of the key science drivers for DOE high-end computation that were identified in the 2005 Greenbook report. We identify any changes in the science drivers and commensurate requirements that might affect the selection of next-generation systems. We then provide an introduction to NERSC's workload analysis and benchmark selection process. Finally, we present an analysis of the performance of the selected benchmark applications. A brief snapshot of benchmark performance as of May 2008 is presented. These are preliminary data, measured, in general, with no attempt at optimization; in other words, representing what a user might observe upon initial porting of the codes. In some cases the data contain estimates, either because the full run is too long or because it requires more cores than we have access to on the given machine.

Overview of the NERSC Workload

NERSC serves a diverse user community of over 3000 users and 400 distinct projects. The workload is comprised of some 600 codes to serve the diverse science needs of the DOE Office of Science research community. Despite the diversity of codes, the median job size at NERSC is a substantial fraction of the overall size of NERSC's flagship XT4 computing system.

NERSC's workload covers the needs of 15 science areas across six DOE Office of Science divisions, in addition to users from other research and academic institutions outside of the DOE. The pie charts in Figure 1 show the percentages of cycles at NERSC awarded to the various DOE offices and science areas.

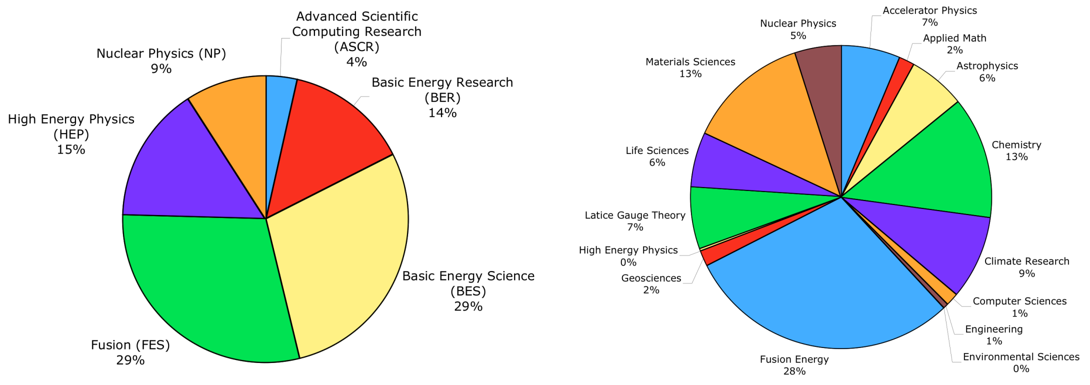


Figure 1: Awards by DOE office (left) and awards by science area (right).

In this study we focused on the largest components of the NERSC workload, which include Fusion, Chemistry, Materials Science, Climate Research, Lattice Gauge Theory, Accelerator Physics, Astrophysics, Life Sciences, and Nuclear Physics, as shown in Figure 2.

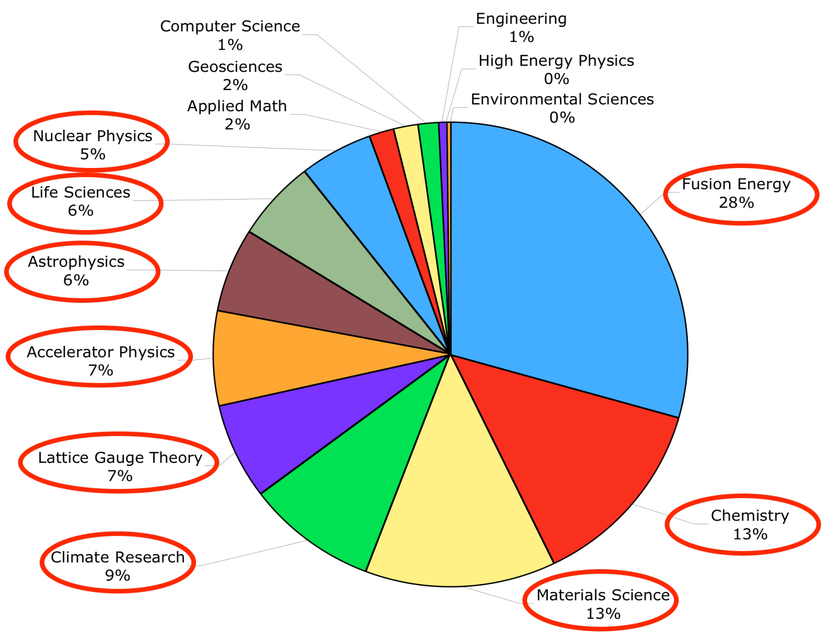


Figure 2: The NERSC workload study focused on dominant contributors to the NERSC workload.

Climate Science

Climate science computing at NERSC includes a variety of prediction and simulation activities, including critical support of the U.S. submission to the Intergovernmental Panel on Climate Change (IPCC). Figure 3 shows that climate science allocations are dominated by CAM, the Community Climate System Model (CCSM3), and weather modeling codes. There are INCITE allocations that are also large players, but because INCITE is not a steady-state allocation we do not consider these projects further. With the INCITE allocations removed, the dominance of CAM and CCSM3 is clear (Table 1). Furthermore, CAM is the dominant computational component of the fully coupled CCSM3 climate model. Therefore, CAM is a good proxy for the most demanding components of the climate workload.

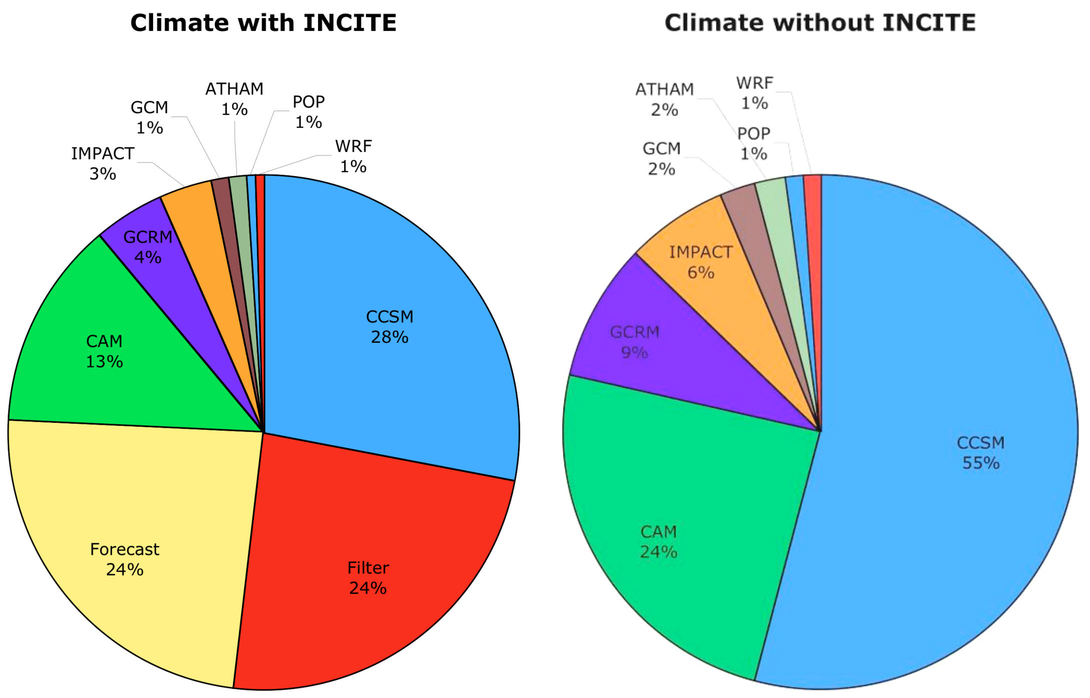


Figure 3: Distribution of climate allocation with and without the contribution of the INCITE allocations. The INCITE awards for "forecast" and "filter" codes are much larger than they are in the steady-state workload

	Code	MPP Award	Percent	Cumulative%
1	CCSM	2,342,000	51%	51%
2	CAM	2,000,000	23%	74%
3	GCRM	2,000,000	8%	82%
4	IMPACT	1,085,000	6%	88%
5	GCM	375,000	2%	90%
6	ATHAM	280,000	2%	92%
7	POP	100,000	1%	93%
8	WRF	80,000	1%	94%

Table 1: Breakdown of the NERSC workload in Climate Sciences. CCSM and CAM account for > 74% of the workload. Since CAM is the dominant component of CCSM, CAM is clearly a good proxy for the climate requirements.

In order to be useful for climate research, CAM must achieve a target performance of 1000x faster than real-time. This places a stringent requirement for the achieved performance of the climate application on the target system architecture. The current mesh resolution ultimately limits the exposable parallelism of the application. However, if the resolution is increased, the size of the simulation time steps must be reduced to satisfy the Courant stability condition. Increasing the resolution by 2x requires 4x more time due to the shorter time steps. Therefore, the climate application pushes towards higher per-processor sustained performance, which runs counter to current microprocessor architectural trends.

Although spectral CAM dominates the current workload, the IPCC is rapidly migrating to the finite-volume formulation of CAM (fvCAM). Finite Volume CAM is more scalable than spectral CAM and is better able to exploit loop level parallelism using a hybrid OpenMP+MPI programming mode and therefore, take advantage of SMP nodes. The hybrid model may mitigate the need for improved per-processor serial performance but would require wider SMP nodes.

In anticipation of the larger mesh resolution targets, we selected the D-mesh (~0.5 degree), which is likely to be the production resolution for some time.

Materials Science

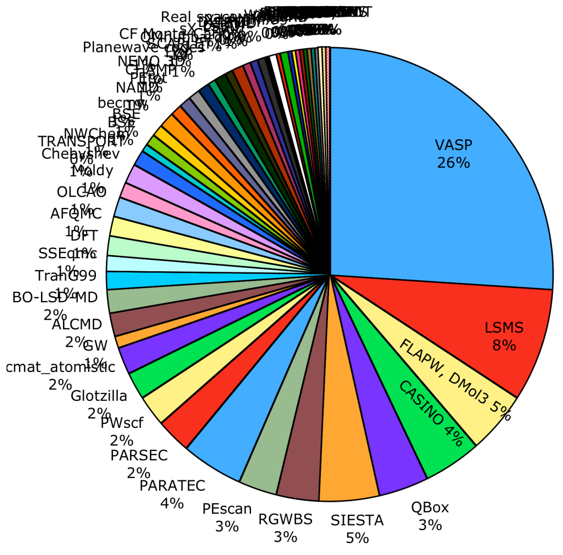
Simulations in material science play an indispensable role in nanoscience research, an area identified as one of the seven highest priorities for the Office of Science. Of particular interest recently are studies of Quantum Dots, due to their emerging importance in miniaturization of electronic and optoelectronic devices, and computational needs seem virtually unlimited. For example, two specific applications that scientists in this area foresee include the following:

- Electronic structures and magnetic materials. The current state of the art is a 500-atom structure, which requires 1.0 Tflop/s sustained performance (typically for several hours), and 100 GB main memory. Future requirements (a hard disk drive simulation, for instance) are for a 5000-atom structure, which will require roughly 30 Tflop/s and 4 TB main memory.

- Molecular dynamics calculations. The current state-of-the-art is for a 109-atom structure, which requires 1 Tflop/s sustained and 50 GB memory. Future requirements (an alloy microstructure analysis) will require 20 Tflop/s sustained and 5 TB main memory.

The materials workload presents a dizzying array of codes, as shown in Figure 4. There are 65 accounts under the BES/material science category that account for 20% of the 324 total NERSC accounts. The total HPC allocation for these accounts was 4.1 M hours before Bassi was online and 8.8 M hours after the Bassi was online. Although this allocation as a function of the total is smaller than it was a few years ago, it still represents 13% of the total NERSC allocation (66.7 M hours after Bassi was online). The materials science percentage has decreased partly because of the creations of special programs like SciDAC and INCITE.

In this science area there is an average of about one code per account (or group of users). However, since the same code can be used by different accounts, one code is used by 2.15 user groups on average. Except for the VASP code, which is used by 23 groups, the majority of the codes are used by less than 5 groups, and many of the codes are used only by one group. This is a very diverse community, with many groups using their own codes.



	Code	MPP Hours	Percent	Cumulative%
1	VASP	1,992,110	26%	26%
2	LSMS	600,000	8%	34%
3	FLAPW, DMol3	350,000	5%	39%
4	CASINO	312,500	4%	43%
5	QBox	262,500	3%	46%
6	SIESTA	346,500	5%	51%
7	RGWBS	232,500	3%	54%
8	PEscan	220,000	3%	57%
9	PARATEC	337,500	4%	61%
10	PARSEC	182,500	2%	64%
	Other	167,300	34%	66%

Figure 4: The materials science area presents a dizzying array of codes, none of which dominate the workload.

The different codes can be classified into six categories based on their physics and the corresponding algorithms. These categories, and their corresponding allocations, are: density functional theory (DFT, 74%); beyond DFT (GW+BSE, 6.9%); quantum Monte Carlo (QMC, 6.7%); classical molecular dynamics (CMD, 6.4%); (classical Monte Carlo(CMC, 3.1%); and other partial differential equations (PDE, 2.9%). There is an overwhelming preference for the DFT method, owing to the current success of that method in ab initio materials science simulations. However, the DFT method itself has different numerical approaches such as plane wave DFT, Green's function DFT, localized basis and orbital DFT, muffin tin sphere type DFT, and real space grid DFT. The most popular (both in terms of number of codes and HPC hours) is plane wave DFT, for which there are 12 codes accounting for 1.6 M hours.

Density Functional Theory (DFT)

In the DFT method, one needs to solve the single-particle Schrödinger equation (a second order partial differential equation). Typically, 5–10% of the lowest eigenvectors are needed from this solution, and the number of eigenvectors is proportional to the number of electrons in the system. For example, for a thousand-atom system, a few thousand eigenvectors are needed. There is a key difference between the complexity of this approach and that of most engineering problems (e.g, fluid dynamics, climate simulation, combustion, electromagnetics) where only a small, fixed number of time evolving or eigenvector fields are solved. In DFT, the Schrödinger equation itself depends on the eigenvectors through the density function (which gives rise to the name density functional theory). Thus, it is a nonlinear problem, but one that can be solved by self-consistent iteration of the linearized eigenstate represented by Schrödinger's equation, or by direct nonlinear minimization. Currently, most large-scale calculations are done using self-consistent iterations. Numerically, what distinguish the different DFT methods and codes are the different basis sets used to describe the wavefunctions (the eigenvectors).

Planewave DFT

Planewave DFT uses planewaves to describe the wavefunctions, while real space DFT uses a regular real-space grid. Due to the sharp peak of the potential near atomic nuclei, special care is needed to choose different basis sets. Besides the planewave and real-space grid, other conventional basis sets include: atomic orbital basis set, where the eigenvectors of the atomic Schrödinger equation are used to describe the wavefunctions in a solid or molecule; Gaussian basis sets, which are more often used in quantum chemistry due to their analytical properties; muffin-tin basis, where a spherical hole is cast out near each nucleus and spherical harmonics and Bessel functions are used to describe the wavefunction inside the hole; augmented planewaves, where spherical Bessel functions near the nuclei are connected with the planewaves in the interstitial regions and used as the basis set; and the wavelet basis sets. In terms of the methods to solve the eigenstate problem, both iterative scheme and direct eigensolvers have been used in different codes. In the planewave DFT, an iterative method (e.g., conjugate gradient) is often used; while in the atomic orbital, Gaussian, and augmented plane-wave (FLAPW) methods, direct dense solvers (ScaLAPACK) are often used; and in real-space grid methods, sparse matrix solvers are used.

There are three key components of the iterative planewave method: 3D global FFT for converting between real and reciprocal space, orthogonalization of basis functions using dense linear algebra, and pseudopotential calculations using dense linear algebra. The most time consuming steps are the matrix vector multiplication and vector-vector multiplication for orthogonalization. For highly parallel computation, the FFT is the primary bottleneck. However, if the problem size is increased, the computational requirements of the dense linear algebra grow $O(N^3)$ whereas the FFT requirements grow $O(N^2)$. Therefore, weak scaling a DFT problem can produce dramatic improvements in the delivered flop rates given the locality of the linear-algebra calculation; but the benefits of weak-scaling are highly problem dependent. For example, time-domain experiments do not benefit at all from this effect.

GW+BSE

GW+BSE is one approach to calculating excited states and optical spectra. It operates primarily on large, dense matrices. The most time consuming parts are generating and diagonalizing these matrices. The diagonalization is often done using a dense eigensolver (with libraries such as ScaLAPACK). The dimension of the matrix is proportional to the square of the number of electrons in the system.

Quantum Monte Carlo (QMC)

Quantum Monte Carlo (QMC) uses stochastic random walk methods to carry out the multidimensional integration of the many-body wavefunctions. Since it needs an assembly sum of different independent walkers, it is embarrassingly parallel. QMC is a very accurate method, but it suffers from statistical noise; thus, it is difficult to use for atomic dynamics (where the forces on the atoms are needed).

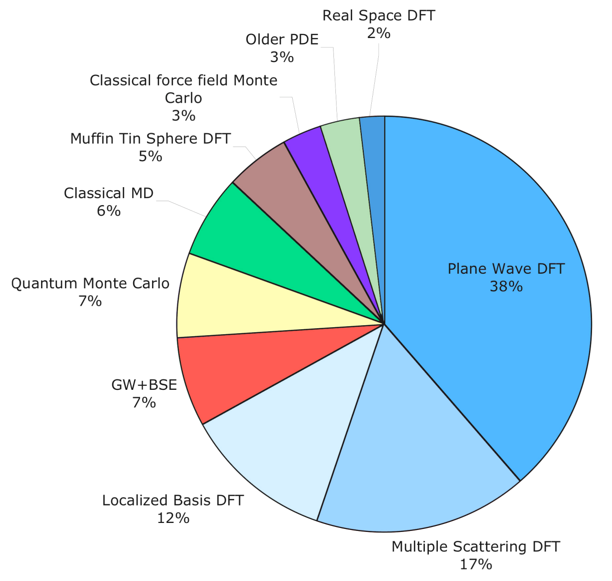
Classical Molecular Dynamics and Classical Monte Carlo

In classical molecular dynamics (MD), the force calculation step is done in parallel. Since the classical force field formalism is local in nature (except the electrostatic force), efficient parallelization is possible as in the NAMD code. Classical Monte Carlo is sometimes used instead of classical MD, thus it is more interested in the time evolving process, instead of an assemble sum (as in quantum MC). As a result, parallelization is not so trivial. There are many recent developments in parallel schemes for classical MC (besides the possible approach for parallel evaluation of the total energy as in classical MD). Other PDEs include Maxwell equations (e.g., in photonic study) and time evolving differential equations for grain boundary and defect dynamics, etc.

Once the material science workload is reorganized using this taxonomy for their computational methods, the distribution of algorithms becomes clearer (Figure 5). The density functional theory (DFT) codes clearly dominate allocations with 72% of the overall workload. Of that subcategory, planewave DFT codes (that include VASP, PARATEC, QBox, and PETot) are the dominant component. The DFT codes (particularly planewave DFT) have very similar computational requirements as previously discussed — with a common set of dominant components in their execution profile including:

- 3D FFT, which requires global communication operations that dominate at high concurrencies and low atom counts
- dense linear algebra for orthogonalization of wave functions (which dominates for large numbers of electronic states in large atomic systems with many electrons but has very localized communication)
- dense linear algebra for calculating the pseudopotential (that also dominates for large numbers of atoms).

Since VASP was not available for public distribution and QBox is export-controlled, we chose PARATEC as the best-available proxy to the requirements of the Materials Science workload. We also get to share effort with the NSF Track-1 and Track-2 procurement benchmarks that have also adopted PARATEC as a proxy to the Materials Science component of their workloads.



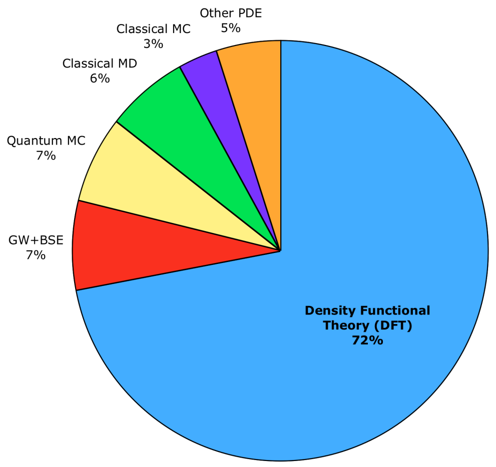


Figure 5: Materials science codes categorized by algorithm.

Chemistry

Despite being one of the most mature simulation fields, the role of computation as a fundamental tool to aid in the understanding of chemical processes continues to grow steadily, and one estimate places the number of electronic structure calculations published yearly about around 8,000!

The dominant single code in the Chemistry workload is S3D due to its substantial INCITE allocation. However, if we set aside INCITE applications because they are not steady-state components of the workload, we see that the chemistry workload is dominated by VASP, which is also dominant in the materials science workload. As a result, we chose to focus on other components of the Chemistry workload to emphasize a different set of criteria in the algorithm space.
S3D bears more similarity to the computational requirements of codes in the astrophysics workload where there are many CFD simulations. We decided to evaluate S3D as a component of that workload simply due to the similarities in the computational methods.

Code	Award	Percent	Cumulative%
ZORI	695,000	12%	12%
MOLPRO	519,024	9%	21%
DACAPO	500,000	9%	29%
GAUSSIAN	408,701	7%	36%
CPMD	396,607	7%	43%
VASP	371,667	6%	49%
GAMESS	364,048	6%	56%

Table 2: Top 50% of the chemistry workload shows a diversity of codes.

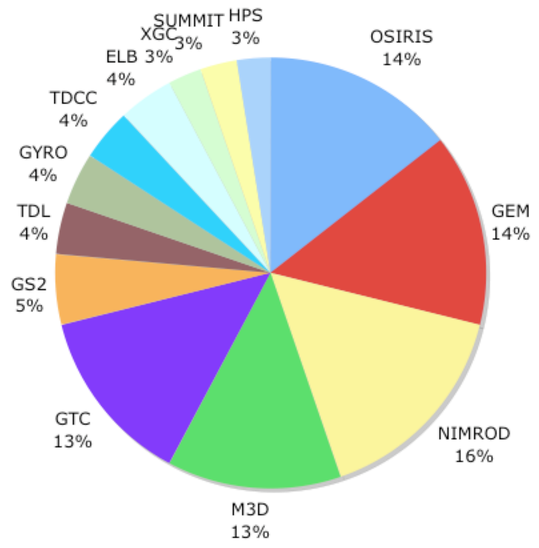
Table 2 shows that the top 50% of the chemistry workload has a diversity of codes. The leader, ZORI, is a QMC code, and as such, emphasizes per-processor performance, but is not very demanding of the interconnect. The remaining codes are either DFT (very similar to materials science) or quantum chemistry codes.

For quantum chemistry the most scalable codes are NWChem and GAMESS, and since these are very demanding of the interconnect they can be useful in differentiating machines based on support for efficient one-sided messaging. GAMESS was selected for NERSC benchmarking. It is also employed by the DOD-MOD TI-0x procurements, which allows us to collaborate with DOD and allows vendors to leverage their benchmarking efforts for multiple system bids.

Fusion

Fusion has long been the dominant component of the NERSC workload, accounting for more than 27% of the overall workload in 2007. The workload is currently dominated by five codes, which account for more than 50% of the overall fusion workload (Table 3). Those codes, OSIRIS, GEM, NIROD, M3D, and GTC, fall into two basic categories. Three of these, OSIRIS, GTC, GEM, are particle-in-cell (PIC) codes, whereas M3D and NIMROD are both continuum codes (PDE solvers on semi-structured grids).

The percentage of per-processor peak achieved by particle-in-cell codes is generally low because the gather/scatter operations required to deposit the charge of each particle on the grid that will be used to calculate the field involve a large number of random accesses to memory, making the code sensitive to memory access latency. GTC is certainly representative in this regard. However, unlike other PIC codes, GTC does not require a global 3-D FFT, so performance and scalability issues associated with this aspect of PIC codes must be covered by benchmarks from other components of the NERSC workload.



code	MPP Award	Percent	Cumulative%
OSIRIS	2,112,500	11%	11%
GEM	2,058,333	11%	22%
NIMROD	2,229,167	12%	34%
M3D	1,921,667	10%	45%
GTC	1,783,333	10%	54%

Table 3: The top 50% of the fusion time usage is dominated by 5 codes.

GTC now utilizes two parallel decompositions, a one-dimensional domain decomposition in the toroidal direction and a particle distribution within each domain. Processors assigned to the same domain have a copy of the local grid but only a fraction of the particles and are linked with each other by a domain-specific communicator used when updating grid quantities. An *Allreduce* operation is required within each domain to sum up the contribution of each processor, which can lead to lower performance in certain cases. Although it's not clear that GTC adequately captures the parallel scaling characteristics of all other PIC codes, it emerges as a clear winning candidate for benchmarking by virtue of its excellent scalability, ease of porting, lack of intellectual property encumbrances, and general simplicity.

On the other hand, we found the fusion continuum codes to be exceedingly complex as benchmarks, one example of which is the required use of external libraries not guaranteed to be present on typical vendor in-house benchmarking machines. However, we also found that the semi-implicit solvers employed by the continuum codes, presumed to be the most time-consuming portion of a run, had similar resource requirements to astrophysics codes that model stellar processes (MHD codes used for modeling core collapse supernovae and astrophysical jets). Therefore, we chose to focus on the PIC codes as proxies for the fusion workload requirements.

Accelerator Modeling

Simulation is playing an increasingly prominent and valuable role in the theory, design and development of particle accelerators, detectors, and X-ray free electron lasers, which are among the largest, most complex scientific instruments in the world.

The accelerator workload is dominated by PIC codes, but also has strong representation by sparse matrix codes with Omega3P (see Table 4).

Code	MPP Award	Percent	Cumulative%
VORPAL	1,529,786	33%	33%
OSIRIS	784,286	16%	49%
QuickPIC	610,000	13%	62%
Omega3p	210,536	4%	66%
Track3p	210,536	4%	70%

Table 4. Top Codes in Accelerator Modeling. The PIC codes VORPAL and OSIRIS dominate the accelerator modeling workload. Even as we look further down the list to 70% of the workload, the allocations are dominated by PIC codes. The exception is Omega3p, which is bottlenecked by its eigensolver, which depends on a sparse-matrix direct solver.

The sparse matrix direct solvers have well known scalability issues that will ultimately force such codes to adopt new algorithmic approaches (likely sparse iterative solvers). There was concern that selection of current sparse matrix code examples would not be reflective of the state-of-the-art computational methods that will emerge for these problems in the 2009 timeframe. Therefore, we concentrated our attention on the PIC component of the workload, which is clearly dominant for this science area.

The PIC codes in accelerator modeling have unique load-balancing problems and potential inter-processor communication issues that distinguish them from plasma simulation PIC codes represented by GTC. Whereas plasma microturbulence codes are generally well load balanced due to the largely uniform distribution of the plasma throughout the computational domain, many accelerator problems depend on bunching up the particles because of the narrow focus of the accelerator beam. This can lead to serious issues for load balancing. Additionally, some of the accelerator codes, notably those comprising the IMPACT suite from LBNL, use an FFT algorithm to solve Poisson's equation, with its associated global, all-to-all communication.

IMPACT-T was chosen as the benchmark representing beam dynamics simulation. Part of a suite of codes used for this purpose, IMPACT-T is used specifically for photoinjectors and linear accelerators. It uses a 2-D domain decomposition of the 3-D Cartesian grid and is one of the few codes used in the photoinjector community that has a parallel implementation. It has an object-oriented Fortran90 coding style that is critical for providing some of its key features. For example, it has a flexible data structure that allows particles to be stored in "containers" with common characteristics such as particle species or photoinjector slices; this allows binning in the space-charge calculation for beams with large energy spreads.

Astrophysics

In the past, NERSC resources have been brought to bear on some of the most significant problems in computational astrophysics, including supernova explosions, thermonuclear supernovae, and analysis of cosmic microwave background data.

Work done by various researchers at LBNL has shown that I/O benchmarking is most effectively done using I/O kernel benchmarks such as IOR and stripped-down applications such as MADCAP rather than using full application codes. This is in part because of the flexibility IOR offers in measuring a range of file and block sizes, I/O methods, and file sharing approaches.

Since I/O benchmarking has essentially been factored out from the main workload, therefore, we concentrate on the other solvers in the astrophysics workload that include a variety of radiation hydrodynamics, MHD, and other astrophysical fluids calculations.

PDE solvers on block-structured grids are a dominant component of this workload. Although many of these codes currently employ an explicit time-stepping scheme, many of the scientists in the field indicate they are moving as quickly as possible towards implicit schemes (such as Newton Krylov schemes) that have more demanding communication requirements but offer a much better time-to-solution.

In particular, many implicit schemes are limited by fast global reductions. Virtually all modern Krylov subspace algorithms (i.e., any thing of the family of conjugate gradient, biconjugate gradient, biconjugate gradient-stabilized, generalized minimum residual, etc. methods) for sparse linear systems require inner products of vectors in order to function. In Fortran, for two vectors X and Y, this would look like

```
inner_prod = 0.0
do i=1,n
inner_prod = inner_prod+X(i)*Y(i)
enddo
```

When the vectors are distributed across a parallel architecture, the "Do" loop sum is only over the elements of the vector that exist on each processor. The global summation is then done over the entire process topology with a local sum on each processor followed by a global reduction operation to get the total sum over the process topology. Thus it would look roughly like:

```
local_sum = 0.0
do i=1,nlocal
local_sum = local_sum+X(i)*Y(i)
enddo
call mpi_all_reduce(local_sum,inner_product,MPI_SUM)
```

Virtually all Krylov subspace algorithms require these kinds of inner product summations. Most classes of implicit algorithms in turn rely heavily on these Krylov subspace algorithms: sparse linear systems, sparse non-linear systems, optimization, sensitivity analysis, etc. Therefore, we felt it necessary to select a benchmark that emphasizes the demands of implicit time stepping schemes to that we don't select a machine that is only good for explicit fluid dynamics algorithms but will ignore the many new applications that utilize implicit approaches that are just starting to become viable.

Unfortunately, at this time very few codes have been developed that fully implement implicit time stepping schemes. There were no clearly dominant codes that used implicit time stepping schemes. Such codes are still in development and are not very portable. We selected the MAESTRO code in part because of close interactions with the developers.

MAESTRO is a low mach number astrophysics code used to simulate the long-time evolution leading up to a type 1a supernova explosion. The NERSC-6 MAESTRO simulation examines convection in a white dwarf and includes both explicit and implicit solvers. MAESTRO is capable of decomposing a uniform grid of constant resolution into a specified number of non-overlapping grid patches but in the NERSC-6 benchmark does not refine the grid.

AMR

Although adaptive mesh refinement (AMR) is a powerful technique with the potential to reduce resources necessary to solve otherwise intractable problems in a variety of computational science fields, it brings with it a variety of additional performance challenges that are worthy of testing in a workload-driven evaluation environment. In particular, they have the potential to add non-uniform memory access and irregular inter-processor communication to what might have otherwise been relatively regular and uniform access and patterns.

We represent AMR in the NERSC-6 benchmark suite with a stripped-down application that exercises an elliptic solver in the Chombo AMR framework developed at LBNL. Chombo provides a distributed infrastructure for implementing finite difference methods for PDEs on block-structured adaptively refined rectangular grids. The benchmark problem uses a solver with a fixed number of iterations and weak scales the problem from 256 to 4096 cores.

For more information see
http://www.nersc.gov/projects/SDSA/software/?benchmark=AMR .

Nuclear Physics

It is no surprise that MILC and other variants of lattice QCD codes dominate the workload.

MILC was also adopted by the NSF procurements as a benchmark, so we are able to leverage the investment in packaging this benchmark for the procurement.

Biology and Life Sciences

Codes in this group seem to fall into three basic categories. One consists of codes for sequence searching, comparison, identification, and/or prediction. Basically informatics types codes, these differ from typical HPC workloads and are not represented to any extent in NERSC benchmark suites. These codes had a moderate allocation in 2007, but we concluded that explicitly representing them in the benchmark suite was unnecessary because the codes are basically constrained by clock speed (integer processing rates, actually) and thus do not present a sufficiently challenging or unique problem to the architecture.

The second category consists of a variety of molecular dynamics codes, the most important of which are AMBER, NAMD, and CHARMM. The capability these codes offer is merged with requirements for protein folding analysis to create what is referred to as molecular dynameomics — one of the most challenging areas of computational science in general. The characteristics of these codes are, in a macro sense, similar to, if not the same as, the molecular dynamics codes used in material science; however, individual life-science related molecular dynamics simulations are likely to be less scalable and more inter-processor communication limited than those of material science, since inter-atomic potentials are likely to be more subtle and longer range.

The third category consists of codes that also are or look like material science or chemistry codes, either for quantum Monte Carlo, classical quantum chemistry, or DFT methods. The computational characteristics of these have already been discussed and likely do not differ for life science applications.

The total life science NERSC allocation in recent years has remained small and so no code has been chosen to uniquely represent this area.

Benchmark Selection

Benchmark selection is an extraordinarily difficult process. As per Ingrid Bucher's 1983 observation, "benchmarks are only useful insofar as they model the workload that will run on a given system." To that end, we use information from our workload analysis to aid in the selection of benchmarks that are a proxy to the anticipated NERSC system requirements. Ideally, the benchmarks must cover the requirements of the target HPC workload, but there are a number of practical considerations that must be taken into account during the selection process.

Coverage: We assess how well the codes cover science areas that dominate the NERSC workload as well as the coverage of the algorithm space. This creates a two-dimensional matrix of computational methods and science areas that must be covered.

Portability: The selected codes must be able to run on a wide variety of systems to allow valid comparisons across architectures. The implementation of the codes must not be too architecture-specific and the "build" systems for these codes must be robust so that they can be used across many system architectures.

Scalability: It is important to emphasize applications that justify scalable HPC resources. Over time, codes that are unable to scale with the resources tend to have a diminishing role in the HPC workload. In selecting scalable codes, we are not selecting those that scale easily — rather those that present some of the most demanding requirements, but are otherwise engineered to expose sufficient parallelism to run on highly parallel systems.

Open Source: Since the benchmarks must be distributed without restriction to vendors for benchmarking purposes, no proprietary or export-controlled code can be used. Only open-source applications that allow us to share snapshots with vendors can be considered.

Availability: Packaging appropriate benchmark problems requires close collaboration with the developer for assistance and support. Without the assistance of the user community, we cannot select appropriate problems for evaluating systems, and hence will be unable to gauge the value of a given system.

First Cut (Primary Secondary Kernel):

Fusion PIC	**GEM (4.9) / GTC (4.4) / XGC (3.7) / SUMMIT (1.5)**
Accelerator modeling	**OSIRIS (2.9) / VORPAL (2.6) / IMPACTZT / BeamB3D / QuickPIC**
Fusion MHD	**M3D / NIMROD / GS2 / GYRO / BOUT / Dynamo**
DFT	**VASP / LSMS / PWscf / CPMD / PARATEC /**
Quantum Chem	**NWChem / SIESTA / CASINO / GAMESS / Gaussian**
Climate	**CAM / Forecast Model / IMPACT**
LG Physics	**MILC / RHMC**
Other astro	**MADCAP / SN1A / VULCAN / SRH3D / FLASH/MAESTRO**
~~**LifeSci**~~	~~**BLAST / Forge / RepeatMasker**~~

Table 5: Matrix of codes for first cut of the selection process. The codes were ranked based on their scores as candidate benchmarks. Codes that could not be widely distributed or were difficult to port were slowly eliminated until the final 6 benchmarks.

With these considerations in mind, the benchmark team selected problems based on their ability to fulfill these attributes and rank them based on their score in each of these areas as candidate benchmark codes. After gaining experience with the codes by building them or through detailed discussions with the developers, we slowly reduced the list down to six target benchmark codes. The final problem sizes and benchmark configurations were determined after the final codes were selected.

These benchmarks are used individually to understand and compare system performance. Although our analysis of the individual application performance has found specific bottlenecks on each system, distilling down to a set of specific hardware features is unlikely to capture the complex interactions between full applications and correspondingly complex system architectures. Therefore the performance of full applications with their associated complex behavior is used to assess the value of systems. Therefore, we feel using a carefully selected set of full applications, as a proxy to full workload performance is better able to encode the DOE Office of Science workload requirements for the purpose of comparing HPC systems.

Benchmark	Science Area	Algorithm Space	Base Case Concurrency	Problem Description	Lang	Libraries
CAM	Climate (BER)	Navier Stokes CFD	56, 240 Strong scaling	D Grid, (~0.5 deg resolution); 240 timesteps	F90	netCDF
GAMESS	Quantum Chem (BES)	Dense linear algebra, DFT	256, 1024 (Same as TI-09)	rms gradient, MP2 gradient	F77	DDI, BLAS
GTC	Fusion (FES)	PIC, finite difference	512, 2048 Weak scaling	100 particles per cell	F90	
IMPACT-T	Accelerator Physics (HEP)	Largely PIC, FFT component	256,1024 Strong scaling	50 particles per cell (currently)	F90	
MAESTRO	Astrophysics (HEP)	Low Mach Hydro; block structured-grid multiphysics	512, 2048 Weak scaling	64 x 64 x 128 gridpts per proc; 10 timesteps	F90	Boxlib
MILC	Lattice Gauge Physics (NP)	Conjugate gradient, sparse matrix; FFT	256, 1024, 8192 Weak scaling	8 x 8 x 8 x 9 Local Grid, ~70,000 iters	C, assem.	
PARATEC	Material Science (BES)	DFT; FFT, BLAS3	256, 1024 Strong scaling	686 Atoms, 1372 bands, 20 iters	F90	Scalapack

Table 6: Final Benchmark Selection Matrix together with problem sizes and concurrencies.

Science Areas	Dense Linear Algebra	Sparse Linear Algebra	Spectral Methods (FFT)s	Particle Methods	Structured Grids	Unstructured or AMR Grids
Accelerator Science		X	X IMPACT	X IMPACT	X IMPACT	
Astrophysics	X	X MAESTRO	X		X MAESTRO	X MAESTRO
Chemistry	X GAMESS	X				
Climate			X CAM		X CAM	
Combustion					X MAESTRO	X AMR Elliptic
Fusion				X GTC	X GTC	
Lattice Gauge		X MILC	X MILC	X MILC	X MILC	
Material Science	X PARATEC		X PARATEC		X PARATEC	

Table 7: Algorithm coverage by the selected SSP algorithms.

Evolution from NERSC-5 Application Benchmarks

We point out that the selected application benchmarks have changed from those used for the NERSC-5 procurement. The following considerations led to these changes:

Workload Evolution: The workload has changed modestly since the NERSC-5 procurement. Two applications from the last benchmark suite (NAMD and MadBENCH) were removed because of their reduced contribution to the workload and were replaced by MAESTRO and IMPACT-T respectively.

Multicore: Power has ended Dennard scaling, so future performance improvements of HPC systems hinge on doubling the number of processor cores every 18 months rather than the historical clock-frequency scaling. We project the next-generation NERSC system will be 3–5x more capable than NERSC-5. Consequently, the concurrency of the benchmarks has increased by 4x over the NERSC-5 suite.

Strong-Scaling: Over the past 15 years, parallel computing has thrived on weak-scaling of problems. However, with the stall in CPU clock frequencies, there is increased pressure to improve strong-scaling performance of algorithms. This is because for time-domain methods, increases in problem resolution must often be accompanied by corresponding decreases in time-step size to satisfy the courant stability condition. Whereas the time-step decreases could be accommodated by performance improvements of individual cores, next generation algorithms will need to improve their step rates using increased parallelism (strong scaling). Consequently, our problem input decks for the NERSC-6 application benchmarks now emphasize strong-scaling requirements.

Implicit Timestepping: Many PDE solvers on block-structured grids employ explicit timestepping methods, which perform well when weak-scaled on massively concurrent systems. However, it is very difficult to improve strong-scaling performance in light of the stall in CPU clock frequencies. This will provide increased pressure to move towards implicit timestepping schemes, which are far more challenging to scale to high concurrencies. Evidence of this trend has been documented in the 2005 Greenbook and subsequent reports from the Fusion Simulation Project (FSP) and other DOE reports. We have coded the requirement into our benchmark suite using MAESTRO, which implements both implicit and explicit timestepping schemes.

AMR/Multiscale Physics: Although it is not a major component of our workload yet, the presence of scalable AMR codes has increased substantially in our workloads. Given the increased emphasis on multi-physics multiscale problems in SciDAC and increased need for strong-scaling, we expect these methods to become increasingly important component of future workloads.

Support for Lightweight/One-Sided Messaging: As systems scale to even larger concurrencies, the performance of the interconnect for very small messages becomes increasingly important. Already, our current workload includes many chemistry applications that depend on Global Arrays (GA) and other libraries that emulate a global shared address space. These code benefit greatly from efficient support of light-weight messaging, including one-sided message constructs. These requirements are represented by the choice of GAMESS for the application suite.

NERSC Composite Benchmarks (Sustained System Performance)

The highest-concurrency runs for these application benchmarks are also used together in the Sustained System Performance (SSP) composite benchmark, which is discussed in a companion document describing both the SSP and ESP tests. The geometric mean of the performance of these benchmarks is used to estimate the likely delivered performance of the evaluated systems for the target NERSC workload. Therefore, this approach fulfills the observation by Ingrid Bucher that "benchmarks are only useful insofar as they model the intended workload." These benchmarks encode our workload requirements, and consequently the SSP metric provides a good metric for estimating the sustained performance on the anticipated workload.

Analysis of NERSC-6 Benchmark Codes

This section provides more detailed descriptions of the NERSC-6 application benchmark codes and their problem sets, along with some observed characteristics of the codes and some preliminary performance data from various systems.

The benchmarks serve three purposes in the procurement project. First, each NERSC-6 benchmark and its respective test problem has been carefully chosen and developed to represent a particular subset and/or specific characteristic of the expected DOE Office of Science workload on the NERSC-6 system. As it has been stated, "For better or for worse, benchmarks shape a field[1]," and thus, a key component of the NERSC workload-driven evaluation strategy is making available real application codes that capture the performance-critical aspects of the workload.

Second, the benchmarks provide vendors the opportunity to provide NERSC with concrete data (in an RFP response) associated with the performance and scalability of the proposed systems on programmatically important applications. We expect that vendor proposal responses will include the results of running NERSC-6 benchmarks on existing hardware and as well as projections to future, currently unavailable systems. In this role, the benchmarks play an essential role in the proposal evaluation process. The NERSC staff are well qualified to evaluate any projections supplied by vendors as a result of having carried out extensive technology assessments of prospective vendor systems for the timeframe of the NERSC-6 project.

Third, the benchmarks will be used as an integral part of the system acceptance test and part of a continuous measurement of hardware and software performance throughout the operational lifetime of the NERSC-6 system.

The NERSC evaluation strategy uses benchmark programs of varying complexity, ranging from kernels and microbenchmarks through stripped-down applications and full applications. Each has its place: we use kernels and microbenchmarks to gain understanding, but applications to evaluate effectiveness and performance.

The remainder of this document will concentrate on the NERSC-6 full application codes. For each of the codes a few key characteristics are presented, most of which have been obtained using the NERSC Integrated Performance Modeling (IPM) tool - an application profiling layer that uses a unique hashing approach to allow a fixed memory footprint and minimal CPU usage. Key characteristics measured include the type and frequency of application-issued MPI calls, buffer sizes utilized for point-to-point and collective communications, the communication topology of each application, and in some cases, a time profile of the interprocessor communications. Another key parameter of a completely different type is the computational intensity, the ratio of the number of floating-point operations executed to the number of memory operations.

[1] Patterson, D., CS252 Lecture Notes, University of California Berkeley, Spring, 1998.

CAM: CCSM Community Atmosphere Model

The Community Atmosphere Model (CAM) is the atmospheric component of the Community Climate System Model (CCSM) developed at NCAR and elsewhere for the weather and climate research communities. Although generally used in production as part of a coupled system in CCSM, CAM can be run as a standalone (uncoupled) model as it is at NERSC. The NERSC-6 benchmark runs CAM version 3.1 at D resolution (about 0.5 degree) using a finite volume (FV) dynamical core. Two problems are run using strong scaling at 56 and 240 cores.

Atmospheric models consist of two principal components, the dynamics and the physics. The dynamics, for which the solver is referred to as the “dynamical core,” are the large-scale part of a model, the atmospheric equations of motion affecting wind, pressure and temperature that are resolved on the underlying grid. The physics are characterized by subgrid-scale processes such as radiation, moisture convection, friction, and boundary layer interactions that are taken into consideration implicitly (via parameterizations).

In CAM3 as it is run at NERSC, the dynamics are solved using an explicit time integration and finite-volume discretization that is local and entirely in physical space. Hydrostatic equilibrium is assumed and a Lagrangian vertical coordinate is used, which together effectively reduce the dimensionality from three to two.

Relationship to NERSC Workload

CAM is used for both short-term weather prediction and as part of a large climate modeling effort to accurately detect and attribute climate change, predict future climate, and engineer mitigation strategies. An example of a “breakthrough” computation involving CAM might be its use in a fully coupled ocean/land/atmosphere/ice climate simulation with 0.125-degree (approximately 12 kilometer) resolution involving an ensemble of eight to ten independent runs. Doubling horizontal resolution in CAM increases computational cost eightfold. The computational cost of CAM in the CCSM, holding resolution constant, has increased 4x since 1996. More computational complexity is coming in the form of, for example, super-parameterizations of moist processes, and ultimately elimination of parameterization and use of grid-based methods.

Parallelization

The solution procedure for the FV dynamical core involves two parts: (1) dynamics and tracer transport within each control volume and (2) remapping of prognostic data from the Lagrangian frame back to the original vertical coordinate. The remapping occurs on a time scale several times longer than that of the transport.

The version of CAM used by NERSC uses a formalism effectively containing two different, two-dimensional domain decompositions. In the first, the dynamics is decomposed over latitude and vertical level. Because this decomposition is inappropriate for the remap phase, and for the vertical integration step for calculation of pressure, for which there exist vertical dependences, a longitude–latitude decomposition is used for these phases of the computation. Optimized transposes move data from the program structures required by one phase to the other. One key to this *yz* to *xy* transpose scheme is having the number of processors in the latitudinal direction be the same for both of the two-dimensional decompositions.

Figure 6 presents the topological connectivity of communication for fv-CAM — each point in the graph indicates processors exchanging messages, and the color indicates the volume of data exchanged. Figures 7 and 8 give an indication of the extent to which different communication primitives are used in the code and their respective times on one system. Figures 9 and 10 give an indication of the sizes of the MPI messages in an actual run on the NERSC Cray XT4 system Franklin.

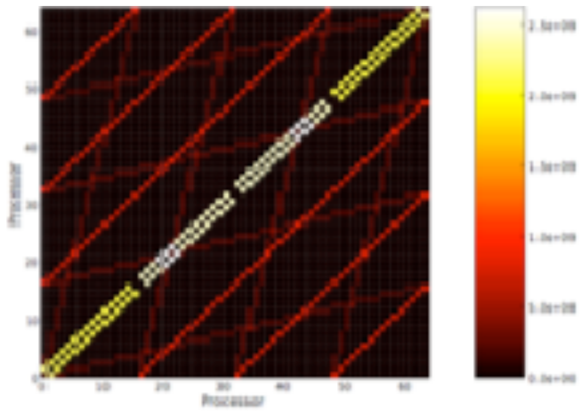


Figure 6. Communication topology and intensity for fvCAM.

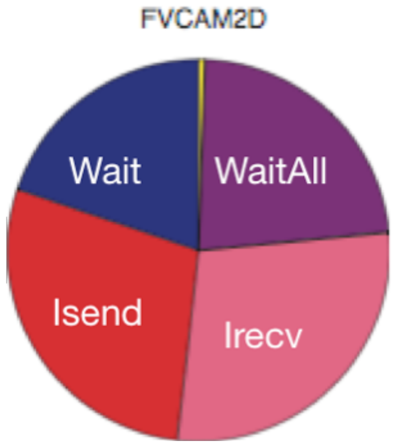


Figure 7. MPI call counts for fvCAM.

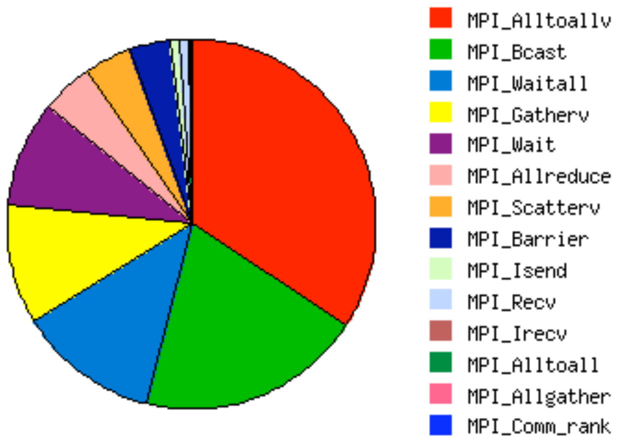


Figure 8. Percent of time spent in various MPI routines for fvCAM on 240 processors on the NERSC Bassi system.

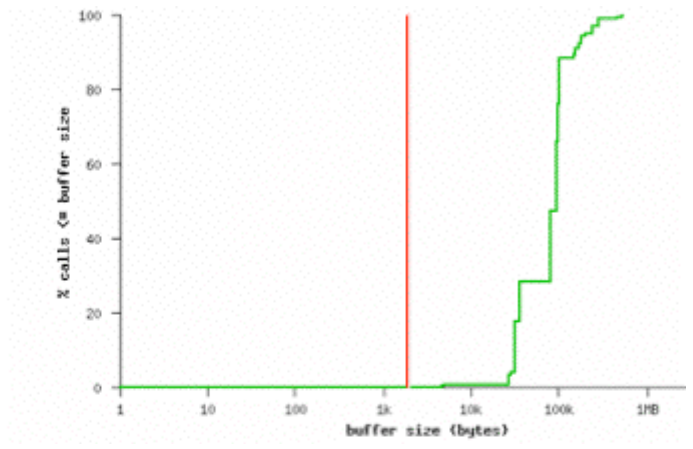


Figure 9. MPI buffer size for point-to-point messages in fvCAM.

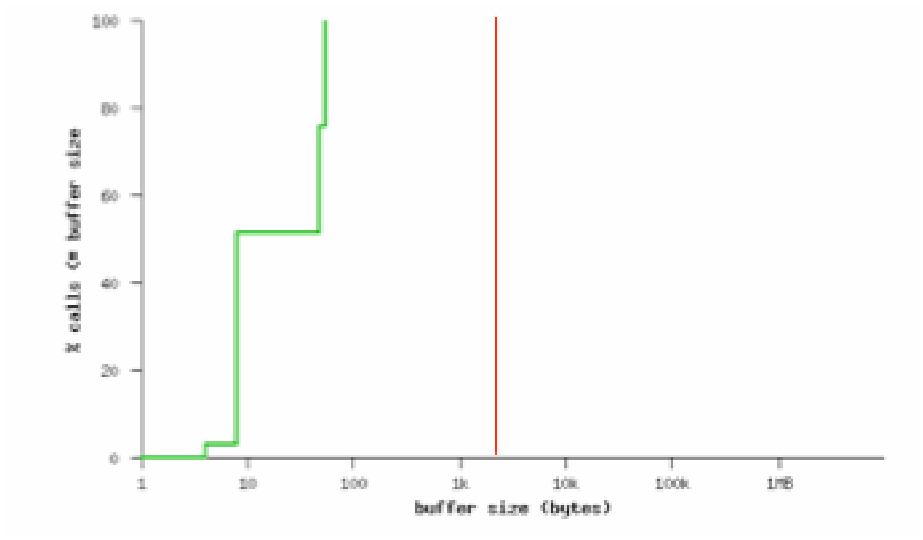


Figure 10. MPI buffer size for collective messages in fvCAM.

Importance for NERSC-6

For NERSC-6 benchmarking, CAM offers the following characteristics: complexity (relatively flat performance profile inhibits simple optimization); relatively low computational intensity stresses on-node/processor data movement; relatively long MPI messages stress interconnect point-to-point bandwidth; strong scaling and decomposition method limit scalability; moderately sensitive to TLB performance and VM page size.

Performance

Table 8 presents measured performance for CAM on several systems. The "GFLOPs" column represents the aggregate computing rate, based on the reference floating-point operation count from Franklin and the time on the specific machine. The efficiency column ("Effic.") represents the percentage of peak performance for the number of cores used to run the benchmark.

Cores	IBM Power5 Bassi		IBM BG/P		QC Opteron Cray XT4 Jaguar		DC Opteron Cray XT4 Franklin	
	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic
56	30	7%	6	3%	32	7%	33.5	12%
240	100	6%	49	3%	143	7%	130.6	11%

Table 8. Measured Aggregate Performance and Percent of Peak for CAM

GAMESS: General Atomic and Molecular Electronic Structure System

The GAMESS (General Atomic and Molecular Electronic Structure System) code from the Gordon research group at the Department of Energy Ames Lab at Iowa State University is one of the most important tools available for various *ab-initio* quantum chemistry simulations. Several kinds of SCF wavefunctions can be computed with configuration interaction, second order perturbation theory, and coupled-cluster approaches, as well as the density functional theory approximation. A variety of molecular properties, ranging from simple dipole moments to frequency-dependent hyper-polarizabilities may be computed. Many basis sets are stored internally, together with effective core potentials, so all elements up to radon may be included in molecules. The two benchmarks used here test DFT energy and gradient, RHF energy, MP2 energy, and MP2 gradient.

Relationship to NERSC Workload

GAMESS is available and is used on all NERSC systems. It is most commonly used by allocations from the BES area. As a benchmark, GAMESS represents a variety of different conventional quantum chemistry codes that use Gaussian basis sets to solve Hartree-Fock, MP2, coupled cluster equations. These codes also now include the capability to perform density functional computations. Codes such as GAMESS can be used to study relatively large molecules, in which case large processor configurations can be used, but frequently they are also used to calculate many-atomic configurations (e.g., to map out the potential manifold in a chemical reaction) for small molecules.

Parallelization

GAMESS uses an SPMD approach but includes its own underlying communication library, called the Distributed Data Interface (DDI), to present the abstraction of a global shared memory with one-side data transfers even on systems with physically distributed memory. In part, the DDI philosophy is to add more processors not just for their compute performance but also to aggregate more total memory. In fact, on systems where the hardware does not contain support for efficient one-sided messages, an MPI implementation of DDI is used in which only one-half of the processors allocated compute while the other half are essentially data movers.

Benchmark Problems

Rather than using GAMESS for scaling studies, two separate runs are done testing different parts of the program. The NERSC “medium” case, run on 64 cores for

NERSC-5 and expected to run on 256 cores for NERSC-6, is a B3LYP(5)/6-311G(d,p) calculation with a RHF SCF gradient. The "large" case, 384 cores for NERSC-5 and expected to run on 1024 cores for NERSC-6, is a 6-311++G(d,p) MP2 computation on a five-atom system with T symmetry.

Importance for NERSC-6

For NERSC-6 benchmarking, GAMESS offers the following characteristics: complexity (relatively flat performance profile inhibits simple optimization but built-in communication layer affords opportunity for key system-dependent optimization); considerable stride-1 memory access; built-in communication layer yields performance characteristics not visible from MPI (typical of Global Array codes, for example).

Performance

Since the problem sets for NERSC-6 GAMESS have not been finalized, the data presented in Table 9 are for the NERSC-5 problems. Performance is not expected to differ significantly.

Cores	**IBM Power5 Bassi**		**IBM BG/P**		**QC Opteron Cray XT4 Jaguar**		**DC Opteron Cray XT4 Franklin**	
	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic
64	26	5%			18	3%	23	7%
384	121	4%	39	2%	177	6%	214	11%

Table 9. Measured Aggregate Performance and Percent of Peak for GAMESS

GTC: 3D Gyrokinetic Toroidal Code

GTC is a three-dimensional code used to study microturbulence in magnetically confined toroidal fusion plasmas via the particle-in-cell (PIC) method. It solves the gyro-averaged Vlasov equation in real space using global gyrokinetic techniques and an electrostatic approximation. The Vlasov equation describes the evolution of a system of particles under the effects of self-consistent electromagnetic fields. The unknown is the flux, f(t,x,v), which is a function of time t, position x, and velocity v, and represents the distribution function of particles (electrons, ions, etc.) in phase space. This model assumes a collisionless plasma, i.e., the particles interact with one another only through a self-consistent field and thus the algorithm scales as N instead of N^2, where N is the number of particles.

A "classical" PIC algorithm involves using the notion of discrete "particles" to sample a charge distribution function. The particles interact via a grid, on which the potential is calculated from charges "deposited" at the grid points. Four steps are involved iteratively:

- "SCATTER," or deposit, charges on the grid. A particle typically contributes to many grid points and many particles contribute to any one grid point.

- Solve the Poisson equation.
- “GATHER” forces on each particle from the resultant potential function.
- “PUSH” (move) the particles to new locations on the grid.

In GTC, point-charge particles are replaced by charged rings using a four-point gyro averaging scheme, as shown in Figure 11.

Relation to NERSC Workload

GTC is used for fusion energy research via the DOE SciDAC program and for the international fusion collaboration. Support for it comes from the DOE Office of Fusion Energy Science. It is used for studying neoclassical and turbulent transport in tokamaks and stellarators as well as for investigating hot-particle physics, toroidal Alfven modes, and neoclassical tearing modes.

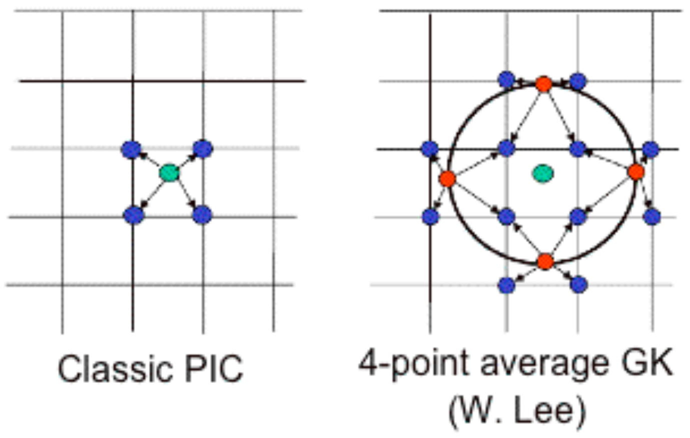


Figure 11. Representation of the charge accumulation to the grid in GTC.

Parallelization

In older versions of GTC, the primary parallel programming model was a 1-D domain decomposition with each MPI process assigned a toroidal section and MPI_SENDRECV used to shift particles moving between domains.

Newer versions of GTC, including the current NERSC-6 benchmark, use a fixed, 1-D domain decomposition with 64 domains plus a particle-based decomposition. At greater than 64 processors, each domain in the 1-D domain decomposition has more than one processor associated with it, and each processor holds a fraction of the total number of particles in that domain. This requires an all_reduce operation to collect all of the charges from all particles in a given domain.

The communication topology of the NERSC-6 version of GTC, using the combination toroidal domain/particle decomposition method, is shown in Figure 12. Figures 13, 14, and 15 show other communication characteristics.

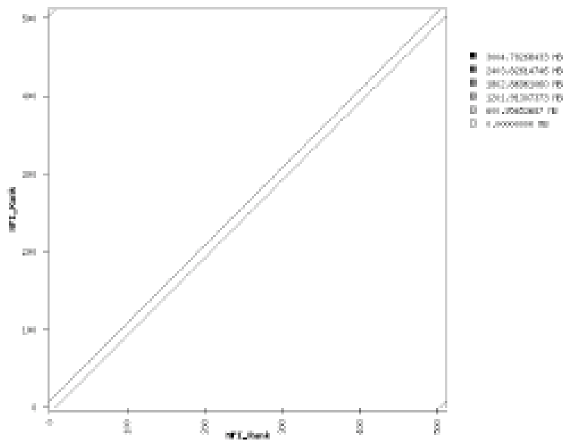


Figure 12. MPI message topology for NERSC-6 version of GTC.

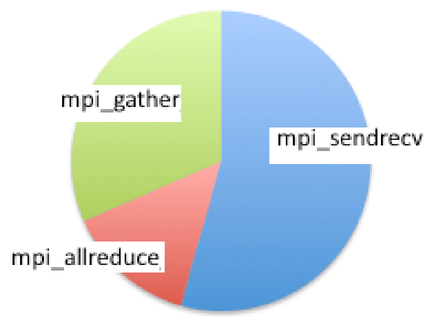


Figure 13. MPI call counts for the NERSC-6 version of GTC.

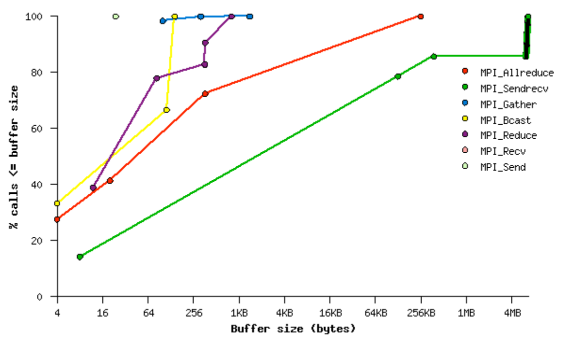


Figure 14. MPI message buffer size distribution based on calls for the NERSC-6 version of GTC.

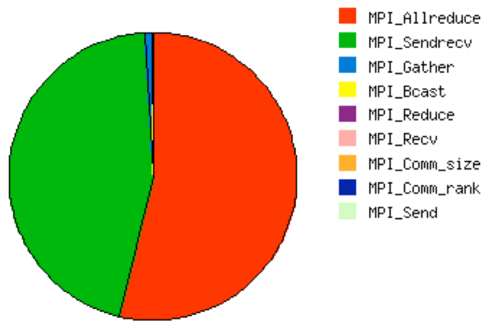


Figure 15. Time spent in MPI functions in the NERSC-6 version of GTC on Franklin.

Importance for NERSC-6

For NERSC-6 benchmarking, GTC offers the following characteristics: charge deposition in PIC codes utilizes indirect addresses and therefore stresses random access to memory; particle-decomposition stresses collective communications with small messages; point-to-point communications are strongly bandwidth bound; runs adequately on relatively simple topology communication networks; important loop constructs with large number of arrays leads to highly sensitive TLB behavior.

Performance

Table 10, which lists some measured performance data, suggests that GTC is capable of achieving relatively high percentages of peak performance on some systems, which is primarily due to a relatively high computational intensity and a low percentage of communication contribution. It also seems that Opteron processors have a performance advantage on this code.

Cores	IBM Power5 Bassi		IBM BG/P		QC Opteron Cray XT4 Jaguar		DC Opteron Cray XT4 Franklin	
	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic
512	428	11%	98	6%	511	12%	565	21%
2048	n/a		391	6%	2023	12%	2438	23%

Table 10. Measured Aggregate Performance and Percent of Peak for GTC.

IMPACT-T: Parallel 3-D Beam Dynamics

IMPACT-T (Integrated Map and Particle Accelerator Tracking — Time) is one member of a suite of computational tools for the prediction and performance enhancement of accelerators. These tools are intended to predict the dynamics of beams in the presence of optical elements and space charge forces, the calculation of electromagnetic modes and wakefields of cavities, the cooling induced by co-moving beams, and the acceleration of beams by intense fields in plasmas generated by beams or lasers. The Impact-T code uses a parallel, relativistic PIC method for simulating 3-D beam dynamics using time as the independent variable. (The design of RF accelerators is normally performed using position, either the arc length or z-direction, rather than the time, as the independent variable.) IMPACT-T includes the arbitrary overlap of fields from a comprehensive set of beamline elements, can model both standing and traveling wave structure, and can include a wide variety of external magnetic focusing elements such as solenoids, dipoles, quadrupoles, etc.

IMPACT-T is unique in the accelerator modeling field in several ways. It uses space-charge solvers based on an integrated Green function to efficiently and accurately treat beams with large aspect ratios, and a shifted Green function to efficiently treat image charge effects of a cathode. It uses energy binning in the space-charge calculation to model beams with large energy spread. It also has a flexible data structure (implemented in object-oriented Fortran) that allows particles to be stored in containers with common characteristics; for photoinjector simulations the containers represent multiple slices, but in other applications they could correspond, e.g., to particles of different species.

While the core PIC method used in IMPACT-T is essentially the same as that used for plasma simulation (see GTC, above), many details of the implementation differ; in particular, the spectral/Green's function potential solver (vs. the finite difference method used in GTC) results in a significant global inter-processor communication pattern.

Relationship to NERSC Workload

Past applications of the IMPACT code suite include the SNS linac modeling, the J-PARC linac commissioning, the RIA driver linac design, the CERN superconducting linac design, the LEDA halo experiment, the BNL photoinjector, and the FNAL NICADD photoinjector. Recent code enhancements have expanded IMPACT's applicability, and it is now being used (along with other codes) on the LCLS project and the FERMI@Elettra project. A recent DOE INCITE grant allowed IMPACT to be used in the design of free electron lasers.

Parallelization

A two-dimensional domain decomposition in the y-z directions is used along with a dynamic load balancing scheme based on domain. The implementation uses MPI-1 message passing. For some problems, most of the communication due to particles crossing processor boundaries is local; however, in the simulation of colliding beams, particles can move a long distance and more than one pair of exchanges might be required for a single particle update. Impact-T does not use a particle-field parallel decomposition used by other PIC codes. Hockney's FFT algorithm is used to solve Poisson's equation with open boundary conditions. In this algorithm, the number of grid points is doubled in each dimension, with the charge density on the original grid kept the same, and the charge density elsewhere is set to zero. Some communication characteristics for IMPACT-T are presented in Tables 11 and 12 and Figures 16–18.

MPI Event	Buffer Size (Bytes)	Cumulative Percent of Total Wall Clock Time
MPI_Alltoallv	131K – 164K	15%
MPI_Send	8K	3%
MPI_Allreduce	4 – 8	4%

Table 11. Important MPI Message Buffer Sizes for IMPACT-T.

Routine Name	Percent of Total Time	Purpose
beambunchclass_kick2t_beambunch	32%	Interpolate the space-charge fields E and B in the lab frame to individual particle + external fields.
beamlineelemclass_getfldt_beamlineelem	29%	Get external field
ptclmgerclass_ptsmv2_ptclmger	10%	Move particles from one processor to four neighboring processors

Table 12. Profile of Top Subroutines in IMPACT-T on Cray XT4

Importance for NERSC-6

For NERSC-6 benchmarking, IMPACT-T offers the following characteristics: object-oriented Fortran90 coding style presents compiler optimization challenges; FFT Poisson solver stresses collective communications with moderate message sizes; greater complexity than other PIC codes due to external fields, open boundary conditions; relatively moderate computational intensity; fixed global problem size causes smaller message sizes and increasing importance of MPI latency at higher concurrencies.

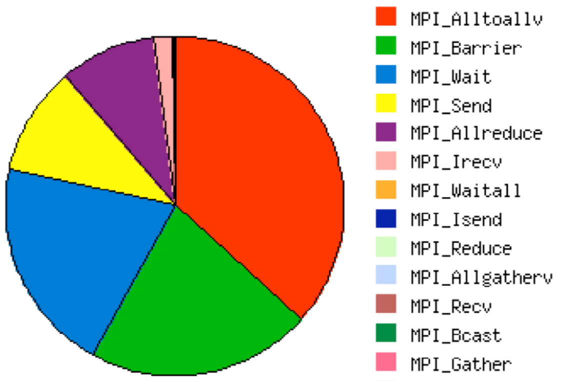


Figure 16. MPI Profile on 1024 cores of Franklin.

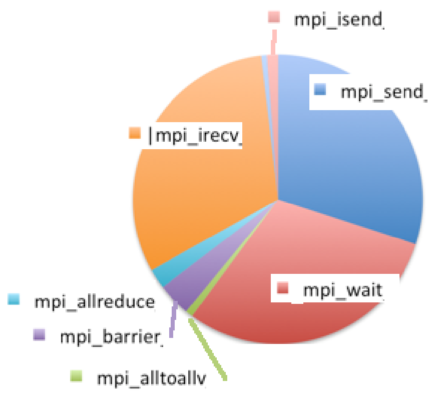


Figure 17. MPI call counts.

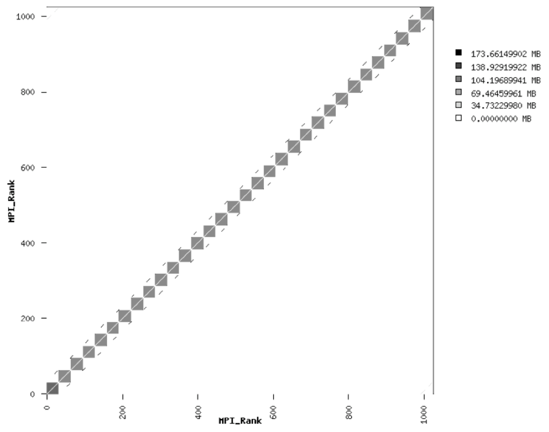


Figure 18. Message topology for IMPACT-T.

Performance

Although they're both PIC codes, performance for IMPACT-T (Table 13) is quite different from that of GTC, with IMPACT-T generally achieving a lower rate, due to increased importance of global communications and a considerably lower computational intensity.

Cores	IBM Power5 Bassi		IBM BG/P		QC Opteron Cray XT4 Jaguar		DC Opteron Cray XT4 Franklin	
	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic
256	143	7%	34	4%	111	5%	130	10%
1024	n/a		174	5%	513	6%	638	12%

Table 13. Measured Aggregate Performance and Percent of Peak for IMPACT-T

MAESTRO: Low Mach Number Stellar Hydrodynamics Code

MAESTRO is used for simulating astrophysical flows such as those leading up to ignition in Type Ia supernovae. It contains a new algorithm specifically designed to neglect acoustic waves in the modeling of low Mach number convective fluids but retain compressibility effects due to nuclear reactions and background stratification that takes place in the centuries-long period prior to explosion. MAESTRO allows for large time steps, finite-amplitude density and temperature perturbations, and the hydrostatic evolution of the base state of the star.

The basic discretization in MAESTRO combines a symmetric operator-split treatment of chemistry and transport with a density-weighted approximate projection method to impose the evolution constraint. The resulting integration proceeds on the time scale of the relatively slow advective transport. Faster diffusion and chemistry processes are treated implicitly in time. This integration scheme is embedded in an adaptive mesh refinement algorithm based on a hierarchical system of rectangular non-overlapping grid patches at multiple levels with different resolution; however, in the NERSC-6 benchmark the grid does not adapt. A multigrid solver is used.

Parallelization

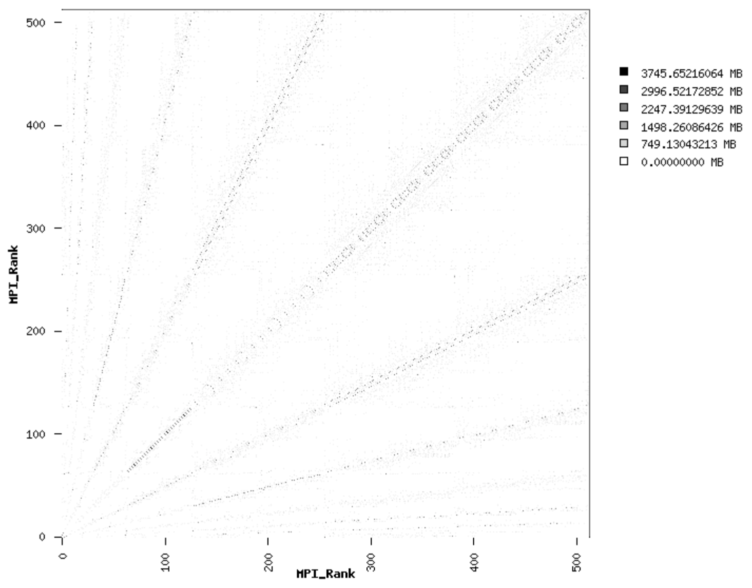
Parallelization in MAESTRO is via a 3-D domain decomposition in which data and work are apportioned using a coarse-grained distribution strategy to balance the load and minimize communication costs. Options for data distribution include a knapsack algorithm, with an additional step to balance communications, and Morton-ordering space-filling curves. Studies have shown that the time-to-solution for the low Mach number algorithm is roughly two orders of magnitude faster than a more traditional compressible reacting flow solver.

MAESTRO uses BoxLib, a foundation library of Fortran90 modules from LBNL that facilitates development of block-structured finite difference algorithms with rich data structures for describing operations that take place on data defined in regions of index space that are unions of non-intersecting rectangles. It is particularly useful in unsteady problems where the regions of interest may change in response to an evolving solution.

The MAESTRO communication topology pattern (Figure 19) is quite unusual. Other communication characteristics are shown in Figures 20–22.

Relationship to NERSC Workload

MAESTRO and the algorithms it represents are used in at least two areas at NERSC: supernovae ignition and turbulent flame combustion studies having both SciDAC, INCITE and base allocations.



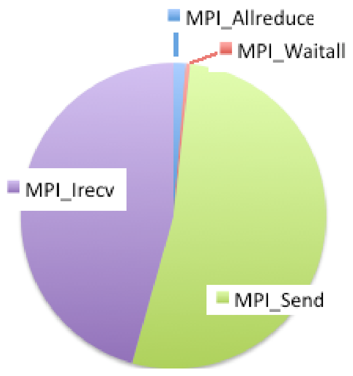
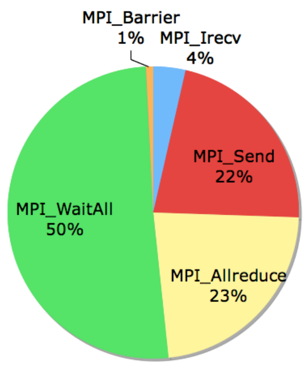


Figure 19. Communication topology for MAESTRO from IPM.

Figure 20. MPI call counts for Maestro



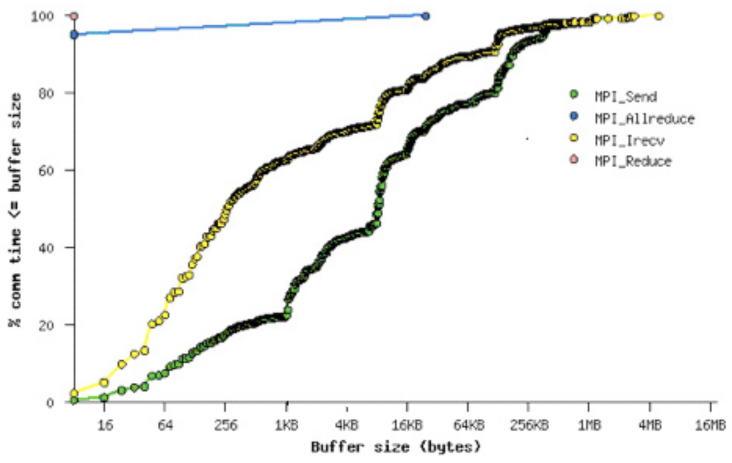


Figure 21. Distribution of MPI times on 512 cores of Franklin from IPM.

Figure 22. Distribution of MPI buffer sizes by time from IPM on Franklin.

Importance for NERSC-6

For NERSC-6 benchmarking, MAESTRO offers the following characteristics: unusual communication topology should stress simple topology interconnects and represent characteristics associated with irregular or refined grids; very low computational intensity stresses memory performance, especially latency; implicit solver technology stresses global communications; wide range of message sizes from short to relatively moderate.

Performance

Maestro achieves a relatively low overall rate on the systems shown in Table 14. The computational intensity is very low, most likely as a result of non-floating-point operation overhead required for the adaptive mesh framework. The data also suggest that MPI communications performance may be a bottleneck, since the efficiency decreases for the larger problem of this weak-scaling set.

Cores	**IBM Power5 Bassi**		**IBM BG/P**		**QC Opteron Cray XT4 Jaguar**		**DC Opteron Cray XT4 Franklin**	
	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic
512	178	5%	52*	3%	230	5%	245	9%
2048	n/a				406	2%	437	4%

Table 14. Measured Aggregate Performance and Percent of Peak for MAESTRO

MILC: MIMD Lattice Computation

The benchmark code MILC represents part of a set of codes written by the MIMD Lattice Computation (MILC) collaboration to study quantum chromodynamics (QCD), the theory of the strong interactions of subatomic physics. It performs simulations of four dimensional SU(3) lattice gauge theory on MIMD parallel machines. Strong interactions are responsible for binding quarks into protons and neutrons and holding them all together in the atomic nucleus.

The MILC collaboration has produced application codes to study several different QCD research areas, only one of which, ks_dynamical simulations with conventional dynamical Kogut-Susskind quarks, is used here.

QCD discretizes space and evaluates field variables on sites and links of a regular hypercube lattice in four-dimensional space time. Each link between nearest neighbors in this lattice is associated with a three-dimensional SU(3) complex matrix for a given field.

QCD involves integrating an equation of motion for hundreds or thousands of time steps that requires inverting a large, sparse matrix at each step of the integration. The sparse matrix problem is solved using a conjugate gradient method, but because the linear system is nearly singular, many CG iterations are required for convergence. Within a processor, the four-dimensional nature of the problem requires gathers from widely separated locations in memory. The matrix in the linear system being solved contains sets of complex three-dimensional “link” matrices, one per 4-D lattice link, but only links between odd sites and even sites are non-zero. The inversion by CG requires repeated three-dimensional complex matrix-vector multiplications, which reduces to a dot product of three pairs of three-dimensional complex vectors. The code separates the real and imaginary parts, producing six dot product pairs of six-dimensional real vectors. Each such dot product consists of five multiply-add operations and one multiply.

Relationship to NERSC Workload

MILC has widespread physics community use and a large allocation of resources on NERSC systems. Funded through High Energy Physics Theory, it supports research that addresses fundamental questions in high energy and nuclear physics and is directly related to major experimental programs in these fields.

Parallelization

The primary parallel programming model for MILC is a 4-D domain decomposition with each MPI process assigned an equal number of sub-lattices of contiguous sites. In a four-dimensional problem, each site has eight nearest neighbors. The code is organized so that message passing routines are compartmentalized from what is built as a library of single-processor linear algebra routines. Many recent production MILC runs have used the RHMC algorithm, which is an improvement in the molecular dynamics evolution process for QCD that reduces computational cost dramatically.

All three NERSC-6 MILC tests are set up to actually do two runs each. This is to more accurately represent the work that the CG solve must do in actual QCD simulations. The CG solver will take insufficient iterations to converge if one starts with an ordered system, so we first do a short run with a few steps, with a larger step size, and with a loose convergence criterion. This lets the lattice evolve from totally ordered. In the NERSC-6 version of the code, this portion of the run is timed as “INIT_TIME” and it takes about two minutes on the NERSC Cray XT4. Then, starting with this “primed” lattice, we increase the accuracy for the CG solve, and the iteration count per solve goes up to a more representative value. This is the portion of the code that we time. Between 65,000 and 75,000 CG iterations are done (depending on problem size).

The three NERSC-6 problems use weak scaling, and the only difference between the three is the size of the lattice; i.e., there are no differences in steps per trajectory or number of trajectories. In Table 15, the local lattice size is based on the target concurrency. Note that these sizes were chosen as a compromise between perceived science needs and a variety of practical considerations. Scaling studies with larger concurrencies can easily be done using MILC. Estimates of QCD runs in the time frame of the NERSC-6 system include some of the lattices in the NERSC-6 benchmark but also up to about 96^3 x 192. Communication characteristics are presented in Figures 23–26.

Target Concurrency	Lattice Size (Global)	Lattice Size (Local)
256 cores	32 x 32 x 32 x 36	8 x 8 x 8 x 9
1024 cores	64 x 64 x 32 x 72	8 x 8 x 8 x 9
8192 cores	64 x 64 x 64 x 144	8 x 8 x 8 x 9

Table 15. NERSC-6 MILC lattice sizes.

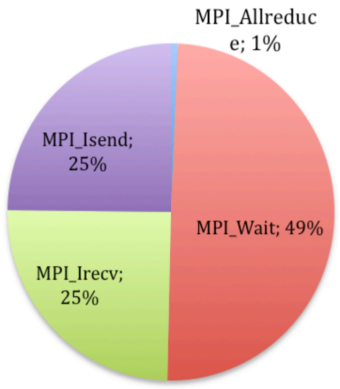


Figure 23. MPI Call counts for MILC.

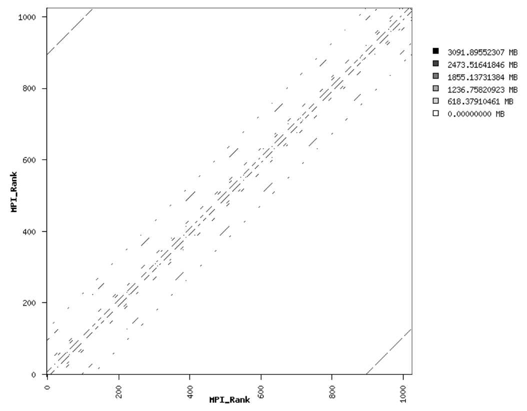


Figure 24. Communication topology for MILC from IPM on Franklin.

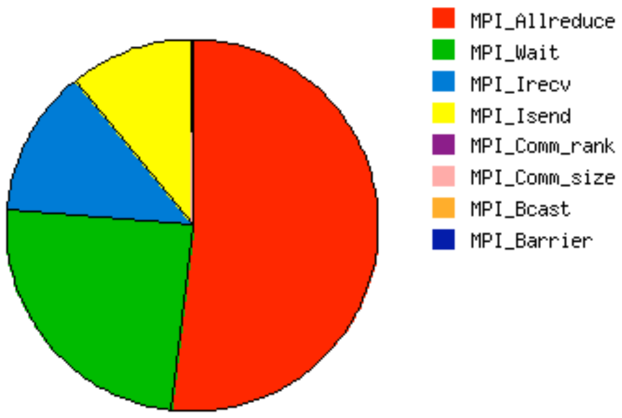


Figure 25. Distribution of Communication times for MILC from IPM on 1024 cores of Franklin.

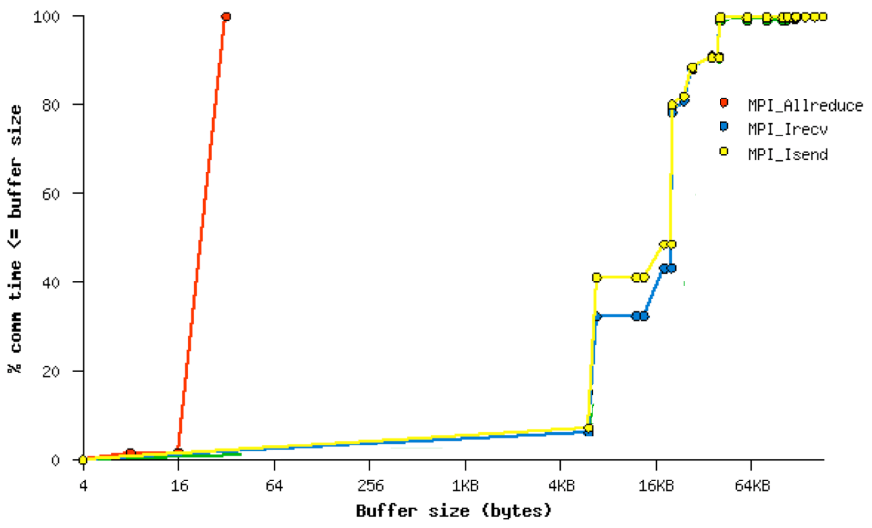


Figure 26. MPI message buffer size distribution based on time for MILC on Franklin from IPM.

Importance for NERSC-6

For NERSC-6 benchmarking, MILC offers the following characteristics: CG solver with small subgrid sizes used stresses interconnect with small messages for both point-to-point and collective operations; extremely dependent on memory bandwidth and prefetching; high computational intensity.

Performance

Table 16 shows that MILC, too, achieves a relatively high percentage of peak on some systems. The dual-core Opteron results include some code tuned for that system, but quad-core Opteron code is not yet available. The increasing effect of communications on larger concurrencies is obvious from the Franklin and Jaguar results on this weak scaled problem.

Cores	**IBM Power5 Bassi**		**IBM BG/P**		**QC Opteron Cray XT4 Jaguar**		**DC Opteron Cray XT4 Franklin**	
	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic
256	488	25%	113	13%	203	9%	291	22%
1024	n/a		456	13%	513	6%	1101	21%
8192	n/a		n/a		3179	5%	5783	14%

Table 16. Measured Aggregate Performance and Percent of Peak for MILC

PARATEC: Parallel Total Energy Code

The benchmark code PARATEC (PARAllel Total Energy Code) performs *ab initio* quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set. Total energy minimization of electrons is done with a self-consistent field (SCF) method. Force calculations are also done to relax the atoms into equilibrium positions. PARATEC uses an all-band (unconstrained) conjugate gradient (CG) approach to solve the Kohn-Sham equations of density functional theory (DFT) and obtain the ground-state electron wavefunctions. In solving the Kohn-Sham equations using a plane wave basis, part of the calculation is carried out in Fourier space, where the wavefunction for each electron is represented by a sphere of points, and the remainder is in real space. Specialized parallel three-dimensional FFTs are used to transform the wavefunctions between real and Fourier space; and a key optimization in PARATEC is to transform only the non-zero grid elements.

The code can use optimized libraries for both basic linear algebra and fast Fourier tranforms; but due to its global communication requirements, architectures with a poor balance between bisection bandwidth and computational rate will suffer performance degradation at higher concurrencies on PARATEC. Nevertheless, due to favorable scaling, high computational intensity and other optimizations, the code generally achieves a high percentage of peak performance on both superscalar and vector systems.

In the benchmark problems supplied here, 20 conjugate gradient iterations are performed; however, a real run might typically do up to 60.

Relationship to NERSC Workload

A recent survey of NERSC ERCAP requests for materials science applications showed that DFT codes similar to PARATEC accounted for nearly 80% of all HPC cycles delivered to the Materials Science community. Supported by DOE BES, PARATEC is an excellent proxy to the application requirements of the entire Materials Science community. PARATEC simulations can also be used to predict nuclear magnetic resonance shifts. The overall goal is to simulate the synthesis and predict the properties of multi-component nanosystems.

Parallelization

In a plane-wave simulation, a grid of points represents each electron's wavefunction. Parallel decomposition can be over n(g), the number of grid points per electron (typically O(100,000) per electron), n(i), the number of electrons (typically O(800) per system simulated), or n(k), a number of sampling points (O(1-10)).

PARATEC uses MPI and parallelizes over grid points, thereby achieving a fine-grain level of parallelism. In Fourier space, each electron's wavefunction grid forms a sphere. Figure 27 depicts a visualization of the parallel data layout on three processors. Each processor holds several columns, which are lines along the z-axis of the FFT grid. Load balancing is important because much of the compute-intensive part of the calculation is carried out in Fourier space. To get good load balancing, the columns are first assigned to processors in descending length order and then to processors containing the fewest points.

The real-space data layout of the wavefunctions is on a standard Cartesian grid, where each processor holds a contiguous part of the space arranged in columns, shown in Figure 27b. Custom three-dimensional FFTs transform between these two data layouts. Data are arranged in columns as the three-dimensional FFT is performed, by taking one-dimensional FFTs along the z, y, and x directions with parallel data transposes between each set of one-dimensional FFTs. These transposes require global inter-processor communication and represent the most important impediment to high scalability. The communication topology for PARATEC is shown below.

The FFT portion of the code scales approximately as $n2\log(n)$, and the dense linear algebra portion scales approximately as n^3; therefore, the overall computation-to-communication ratio scales as n, where n is the number of atoms in the simulation.

In general, the speed at which the FFT proceeds dominates the runtime. Table 19, below, compares the speed of the FFT and BLAS-dominated Projector portions of the code for various machines. The data show that PARATEC is an extremely useful benchmark for discriminating between machines based on both computation and communication speed. Communication characteristics are shown in Figures 28–30 and in Table 17.

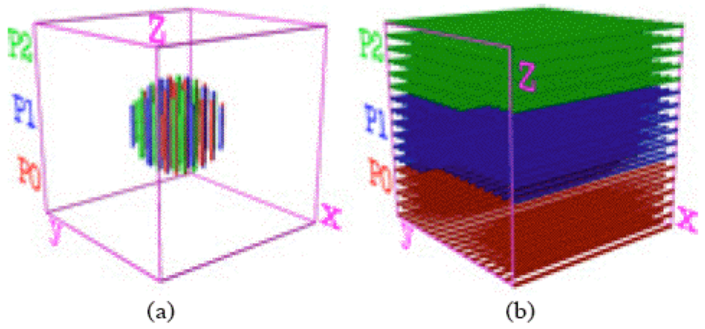


Figure 27. A three-processor example of PARATEC's parallel data layout for the wavefunctions of each electron in (a) Fourier space and (b) real space.

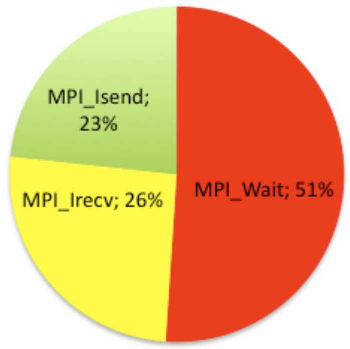


Figure 28. MPI call counts for PARATEC.

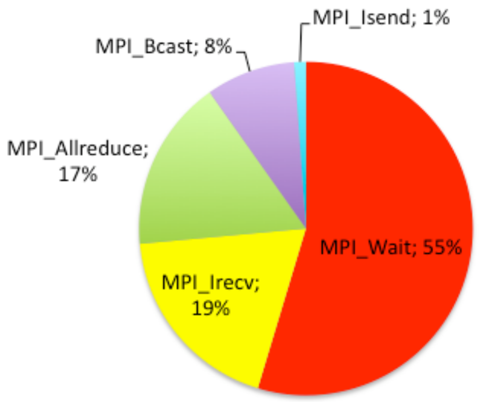


Figure 29. Distribution of MPI times for PARATEC on 256 cores of Franklin from IPM.

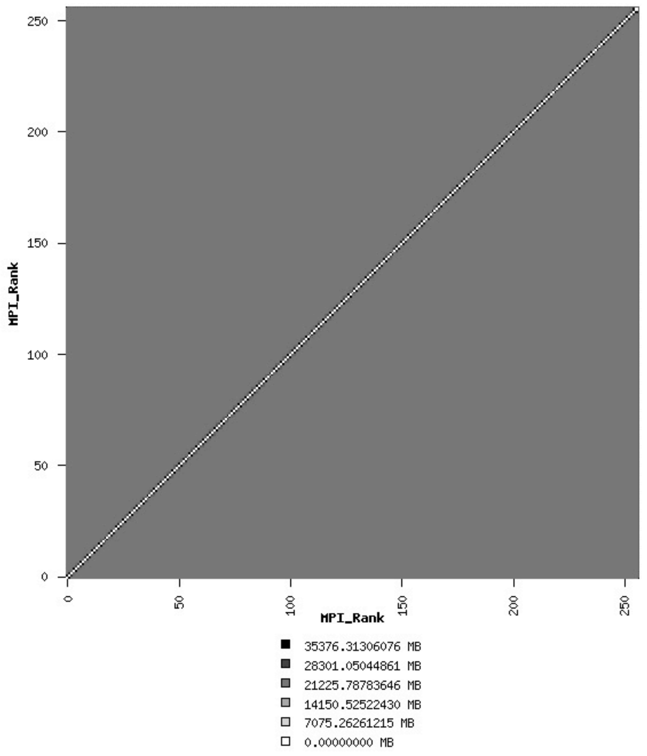


Figure 30. Communication Topology for PARATEC from IPM.

Message Buffer Size	Count on 256 Cores	Count on 1024 Cores
Total Message Count	428,318	1,940,665
16 ≤ MsgSz < 256		114,432
256 ≤ MsgSz < 4 KB	20,337	1,799,211
4 KB ≤ MsgSz < 64 KB	403,917	4,611
64 KB ≤ MsgSz < 1 MB	1,256	22,412
1 MB ≤ MsgSz < 16 MB	2,808	

Table 17. Important MPI Message Buffer Sizes for PARATEC (from IPM on 256 cores, Franklin)

Importance for NERSC-6

For NERSC-6 benchmarking PARATEC offers the following characteristics: hand-coded spectral solver with all-to-all data transpositions stresses global communications bandwidth; mostly point-to-point messages, and since in a single 3-D FFT the size of the data packets scales as the inverse of the inverse of the square of the number of processors, relatively short messages; can be very memory-capacity limited; tuning parameter offers opportunity to trade memory usage for interprocessor communication intensity; requires quality implementation of system libraries (FFTW/SCALAPACK/BLACS/BLAS); use of BLAS3 and 1-D FFT libraries results in high cache reuse and high percentage of per-processor peak performance; fixed global problem size causes smaller message sizes and increasing importance of MPI latency at higher concurrencies; moderately sensitive to TLB performance.

Performance

Tables 18 and 19 show performance data for PARATEC. The code calculates its own floating-point operation rate for two key components, the "Projector" portion of the code, which is dominated by BLAS3 routines, and the FFTs, which use tuned system libraries but depend on global communications. The comparison among systems, especially the dual-core and quad-core Cray XT4 systems, is quite interesting.

Cores	**IBM Power5 Bassi**		**IBM BG/P**		**QC Opteron Cray XT4 Jaguar**		**DC Opteron Cray XT4 Franklin**	
	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic.	GFLOPs	Effic
256	30	7%	517		665	31%	729	55%
1024	n/a		710	20%	2044	24%	2234	42%

Table 18. Measured Aggregate Performance and Percent of Peak for PARATEC

Machine	FFT Rate (MFLOPS/core)	Projector Rate (MFLOPS/core)	Overall Rate (Agg. GFLOPS)
Franklin	213	1045	729
Jaguar	129	1581	665
BG/P	377	561	517
HLRB-II	694	993	890
Bassi	1028	1287	1213

Table 19. Performance Profile for PARATEC (256 cores)

Summary

Table 20 presents a summary of a few of the important characteristics for the codes. In addition to doing a good job of representing key science areas within the NERSC Office of Science workload, the codes provide a good range of architectural "stress points" such as computational intensity, and MPI message intensity, type, and size. As such, the codes provide an effective way of differentiating between systems and will be useful for all three goals related to the NERSC-6 project. Finally, the codes provide a strong suite of SCALABLE benchmarks, capable of being run at a wide range of concurrencies.

	Code						
	CAM	**GAMESS**	**GTC**	**IMPACT-T**	**MAESTRO**	**MILC**	**PARATEC**
CI*	0.67	0.61	1.15	0.77	0.24	1.39	1.50
Cray XT4 % peak per core (largest case)	13%	12%	24%	14%	5%	14%	44%
Cray XT4 % MPI medium	29%		4%	9%	20%	120%	27%
Cray XT4 % MPI large	35%		6%	40%		23%	64%
Cray XT4 % MPI extra large	n/a	n/a	n/a	n/a	n/a	30%	n/a
Cray XT4 avg msg size med	113 K	n/a	1 MB	35 KB	2 K	16 KB	34 KB

Table 20 Comparison of Computational Characteristics for NERSC-6 Benchmarks.
*CI is the computational intensity, the ratio of the number of floating-point operations to the number of memory operations.

Bibliography (partial)

Wong, Adrian T., Leonid Oliker, William T. C. Kramer, Teresa L. Kaltz, and David H. Bailey, "Evaluating System Effectiveness in High Performance Computing Systems," Proceedings of SC2000, November 2000.

Wong, Adrian T., Leonid Oliker, William T. C. Kramer, Teresa L. Kaltz, and David H. Bailey, "System Utilization Benchmark on the Cray T3E and IBM SP," 5th Workshop on Job Scheduling Strategies for Parallel Processing, May 2000, Cancun Mexico.

Kramer, William and Clint Ryan, "Performance Variability on Highly Parallel Architectures", the International Conference on Computational Science 2003, Melbourne Australia and St. Petersburg Russia, June 2-4, 2003.

Ingrid Y. Bucher, "The Computational Speed of Supercomputers," Joint International Conference on Measurement and Modeling of Computer Systems archive Proceedings of the 1983 ACM SIGMETRICS conference on Measurement and modeling of computer systems.

I.Y. Bucher and J.L. Martin, "Methodology for Characterizing a Scientific Workload," In CPEUG'82, pages 121--126, 1982.

Horst D. Simon, William T. C. Kramer, David H. Bailey, Michael J. Banda, E. Wes Bethel, Jonathon T. Carter, James M. Craw, William J. Fortney, John A. Hules, Nancy L. Meyer, Juan C. Meza, Esmond G. Ng, Lynn E. Rippe, William C. Saphir, Francesca Verdier, Howard A. Walter, Katherine A. Yelick, "Science-Driven Computing: NERSC's Plan for 2006–2010," Lawrence Berkeley National Laboratory Report LBNL-57582, May, 2005.

Horst Simon, William Kramer, William Saphir, John Shalf, David Bailey, Leonid Oliker, Michael Banda, C. William McCurdy, John Hules, Andrew Canning, Marc Day, Philip Colella, David Serafini, Michael Wehner and Peter Nugent, "Science-Driven System Architecture: A New Process for Leadership Class Computing," J. Earth Sim., Vol. 2, January 2005.

Leonid Oliker, Andrew Canning, Jonathan Carter, Costin Iancu, Michael Lijewski, Shoaib Kamil, John Shalf, Hongzhang Shan, Erich Strohmaier, Stephane Ethier, Tom Goodale, "Scientific Application Performance on Candidate PetaScale Platforms," International Parallel & Distributed Processing Symposium (IPDPS), March 24-30, 2007, Long Beach, CA.

Leonid Oliker, Andrew Canning, Jonathan Carter, John Shalf, David Skinner, Stephane Ethier, Rupak Biswas, Jahed Djomehri, and Rob Van der Wijngaart, "Performance Evaluation of the SX6 Vector Architecture for Scientific Computations," http://crd.lbl.gov/~oliker/drafts/SX6_eval.pdf

A. S. Almgren, J. B. Bell, M. Zingale, "MAESTRO: A Low Mach Number Stellar Hydrodynamics Code," Journal of Physics: Conference Series 78 (2007) 012085.

J. B. Bell, A. J. Aspden, M. S. Day, M. J. Lijewski, "Numerical simulation of low Mach number reacting flows," Journal of Physics: Conference Series 78 (2007) 012004.

Y. Ding, Z. Huang, C. Limborg-Deprey, J. Qiang, "LCLS Beam Dynamics Studies with the 3-D Parallel Impact-T Code," Stanford Linear Accelerator Center Publication SLAC-PUB-12673, Presented at the 22nd Particle Accelerator Conference, Albuquerque, New Mexico, U.S.A, June 25-29, 2007.

Ji Qiang, Robert D. Ryne, Salman Habib, and Viktor Decyk, "An Object-Oriented Parallel Particle-in-Cell Code for Beam Dynamics Simulation in Linear Accelerators," Journal of Computational Physics 163, 434–451 (2000).

S. C. Jardin, Editor, "DOE Greenbook: Needs and Directions in High Performance Computing for the Office of Science," Princeton Plasma Physics Laboratory Report PPPL-4090, June, 2005. Available from http://www.nersc.gov/news/greenbook/.

William Kramer, John Shalf, and Erich Strohmaier, "The NERSC Sustained System Performance (SSP) Metric," Lawrence Berkeley National Laboratory Report LBNL-58868, http://repositories.cdlib.org/lbnl/LBNL-58868, 2005.

L. Oliker, J. Shalf, J. Carter, A. Canning, S. Kamil, M. Lijewski, S. Ethier (2007). " Performance Characteristics of Potential Petascale Scientific Applications." Petascale Computing: Algorithms and Applications (David Bader: editor), Volume . Chapman & Hall / CRC Press: 2007.

Lin-Wang Wang (2005), "A Survey of Codes and Algorithms used in NERSC Materials Science Allocations," Lawrence Berkeley National Laboratory Report LBNL- 61051, 2005.

A Mirin and P Worley, "Extending Scalability of the Community Atmosphere Model," Journal of Physics: Conference Series 78 (2007) 012082.

Arnold Kritz, David Keyes, et al., "Fusion Simulation Project Workshop Report," available from http://www.ofes.fusion.doe.gov/programdocuments.shtml.

K. Yelick, "Programming in UPC (Unified Parallel C), CS267 Lecture, Spring 2006," lecture notes available from http://upc.lbl.gov/publications/.

For more information on the benchmarks see
http://www.nersc.gov/projects/SDSA/software/?benchmark=NERSC5