Lawrence Berkeley National Laboratory

Lawrence Berkeley National Laboratory

Title

The Performance Effect of Multi-core on Scientific Applications

Permalink

https://escholarship.org/uc/item/7tb957z8

Authors

Carter, Jonathan
He, Yun
Shalf, John
et al.

Publication Date

2007-05-14

eScholarship.org Powered by the California Digital Library
University of California

The Performance Effect of Multi-core on Scientific Applications

Jonathan Carter, Yun (Helen) He, John Shalf,
Hongzhang Shan, Erich Strohmaier, and Harvey Wasserman
NERSC Division, Lawrence Berkeley National Laboratory,
Berkeley, CA 94720

ABSTRACT: *The historical trend of increasing single CPU performance has given way to roadmap of increasing core count. The challenge of effectively utilizing these multi-core chips is just starting to be explored by vendors and application developers alike. In this study, we present some performance measurements of several complete scientific applications on single and dual core Cray XT3 and XT4 systems with a view to characterizing the effects of switching to multi-core chips. We consider effects within a node by using applications run at low concurrencies, and also effects on node-interconnect interaction using higher concurrency results. Finally, we construct a simple performance model based on the principle on-chip shared resource—memory bandwidth—and use this to predict the performance of the forthcoming quad-core system.*

KEYWORDS: Multi-core, scientific computing

1 Introduction

For the past 15 years, CPU performance has improved at an exponential pace – doubling approximately every 18 months with remarkable consistency. In order to maintain performance improvements within the conservative power envelope allowed by practical system design, the historical trend of increasing clock rates at an exponential pace has given way to a chip-scale multiprocessor (CMP) design strategy where the performance of individual CPU cores stays constant and the number of cores increases at an exponential pace [1,2,3].

In the High-End Computing (HEC) arena, the first exposure to CMPs occurred via the Power4 dual-cores from IBM in 2002. In 2005 both Intel and AMD launched commodity dual-core chips with quad-core systems following or to be released later this year. The AMD dual and quad core design are the building blocks of the next generation Cray XT system, the platform selected by both the NERSC at Lawrence Berkeley National Laboratory and the NCCS at Oak Ridge National Laboratory. Both these systems consist of tens of thousands of cores, and we can expect future systems to be composed of millions [4].

Taken as a whole, the transition to multi-core technologies represents a paradigm shift of similar magnitude to that of transitioning from vector supercomputers to massively parallel processor (MPP) machines that occurred in the mid-1990s. While the challenges presented by the shift to MPPs could be summarized as how best to map data and computation onto local and remote resources, the challenges faced in the transition to multi-core arise form the problem of how best to share resources within a CMP.

Initially, both vendors and application writers are treating multi-core chips simply as conventional symmetric multiprocessors, or in the case of the current Cray XT running the Catamount OS, as hosts to multiple tasks running under a SPMD programming model. Leaving for future research the topic of whether and how some of the new features of multi-core

chips can be exploited by alternate programming models, we focus on how the SPMD applications of today are impacted by this transition.

In the following study we examine the performance of several applications drawn from different scientific disciplines on Cray XT3 and XT4 platforms and compare run times and other performance metrics on both single and dual core systems. We extend our previous work [5] by: testing a conjecture regarding the interplay of tuning applications for a single core chip and then moving to a dual core one; by examining how increased concurrency also plays into the multi-core penalty; and by validating our multi-core performance prediction model with a more complete set of applications.

2 Cray XT System Architecture

The basic building block of the Cray XT4 and Cray XT3 systems is a processing element. Each PE is comprised of one AMD Opteron processor (single, dual or soon to be available quad core) coupled with its own memory and dedicated communication resource.

Each PE can be supplied with 1~8 GB of memory, and depending on the AMD socket characteristics this may be DDR1 or DDR2, with different memory speeds also available.

Each Opteron processor is directly connected to the Cray XT interconnect via a Cray SeaStar routing and communications chip over a 6.4 GB/s HyperTransport path. The SeaStar chip acts as the gateway to the Cray XT high bandwidth, low latency interconnect. The router in the SeaStar chip provides six high speed network links, 7.6 GB/s bi-directional, to connect to six neighbors in the 3D torus topology. While the peak bandwidth of the Seastar chips is the same in both XT3 and XT4, the second-generation chip (Seastar 2.1) is capable of twice the injection bandwidth and has a lower message-injection latency.

All the experiments in this study were performed on the NCCS Jaguar system at ORNL. During the past two years, Jaguar has been variously configured as a single-core XT3, dual-core XT3, and most recently as a dual-core hybrid XT3/XT4 system. Most data were collected from its merged XT3/XT4 system, which was completed at the end of March 2007.

3 MILC

The benchmark code MILC [6] 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. Strong interactions are responsible for binding quarks into protons and neutrons and holding them all together in the atomic nucleus. MILC performs simulations of four-dimensional *SU*(3) lattice gauge theory on MIMD parallel machines. The test case data shown here is for a lattice size of 32^4 with two trajectories of five steps each.

In a previous paper [5] we postulated that different degrees of optimization of a given application might change the extent to which the application is affected by multi-core effects. With MILC we have the opportunity to test this hypothesis. We can also study the benefits provided by a faster memory subsystem by comparing MILC performance between the XT3 and XT4.

MILC is a highly memory intensive code and a version that includes a variety of compiler and hand-written optimizations for improving memory throughput has been developed by Cray engineers. The original version of the code simply used the “-O3” compiler optimization flag with gcc; the Cray-supplied version added the compiler options “-funroll-loops -fprefetch-loop-arrays -fomit-frame-pointer” (of which the most important option by far is “-fprefetch-loop-arrays”). The optimized version also included replacement of 15 C routines with inline assembler kernels, and added several

software prefetching macros to loops in two routines.

Using these two versions of the code we now compare times for 64-processor runs under several conditions: XT3 vs. XT4, unoptimized vs. optimized, and single-core vs. dual-core. Data are collected in Tables 3.1 and 3.2.

MILC Version	**Times (seconds)**	
	XT3	**XT4**
Single Core Orig	274	230
Single Core Opt	160	127
Dual Core Orig	358	277
Dual Core Opt	230	181
	Dual Core Penalty	
	XT3	**XT4**
Original	1.31	1.20
Optimized	1.44	1.43

Table 3.1 MILC Timings on Jaguar.

MILC Version	**Improvement: XT4/XT3**	
Single Core Orig	1.19	
Single Core Opt	1.26	
Dual Core Orig	1.29	
Dual Core Opt	1.27	
	Improvement: Optimized / Original	
	XT3	**XT4**
Single Core	1.71	1.81
Dual Core	1.56	1.53

Table 3.2 MILC Performance Ratios on Jaguar.

Based on the data, we can make the following observations. The dual-core penalty is clearly worse for the optimized version of the code than for the original version. This is reasonable: the highly optimized code will make more efficient use of the memory channel in single-core mode; thus, it will experience more contention for that channel in dual-core mode.

An additional, intermediate level of optimization, using C code with prefetch but no SSE, was also examined. Although not shown in the tables, the results are completely consistent with those shown above; i.e., as the level of optimization increases the dual core penalty increases. For this code, using both cores of the dual-core processor greatly reduces the improvement afforded by code optimization. On the dual-core XT4, the optimized code is about 1.5 times faster, but using only a single core on the XT4 shows a speedup of 1.8.

The primary improvement for single-node performance of the XT4 vs. the XT3 is the near doubling of memory bandwidth, which is reflected in the STREAM benchmark that show improvements ranging from 1.4 to 1.8 depending on the function. At first glance the MILC data seem inconsistent with this observation. In dual-core mode the XT4 provides the same improvement in performance for the un-optimized and optimized versions of the code, but in single-core mode the XT4 provides less benefit for the original version – the opposite of what might be expected. Our theory is that a single task of un-optimized MILC application is not capable of saturating the memory interface of the XT4 and does not gain the full benefit of the improved memory bandwidth. Consistent with this theory is the fact that neither version of MILC achieves an XT4-to-XT3 speedup anywhere close to the STREAM improvements.

Turning towards performance effects on high concurrency simulations, Figure 3.1, below, shows the results of a weak scaling study of MILC on Jaguar in XT4 mode, where it can be seen that, for an equivalent number of total cores, the un-optimized version of the code in single-core mode actually runs slightly faster than the optimized version running in dual-core mode above 1000 cores. For the un-optimized version of the code the dual-core penalty is about 20% on 64 cores, rising to about 35% on

4096 cores. For the optimized version the dual-core penalty is about 40% on 64 cores, rising to about 58% on 4096 cores.

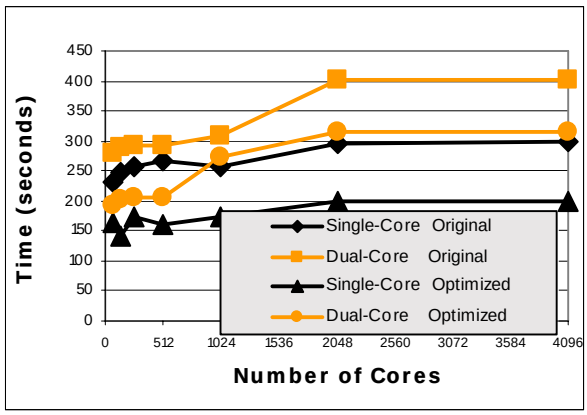

Figure 3.1 Weak Scaling of MILC on Jaguar XT4.

At 64 cores the percentage of time spent in communication is only about 5% of the total, so the impact of dual-core is confined mainly to sharing of main memory bandwidth. At higher concurrencies this is not the case. At the simplest level, the impact can be divided into effects on latency and bandwidth. The CG solver in the MILC code is very sensitive to communication system latency. The measured communication latencies increase considerably when moving from single core to dual core. For example, we have measured an MPI unidirectional inter-node latency of about 4.8 microseconds on Jaguar in single-core mode, but this increases to about 6.3 microseconds when both cores on the two nodes perform the ping experiment (Figure 3.2). This increase in latency is a result of the master-slave relationship between cores on a node. Perhaps even more important is the reduction in unidirectional bandwidth observed (Figure 3.3) as both cores attempt to access the SeaStar channnel: the observed bandwidth for 64K message sizes (typical of MILC) is reduced by about a factor of two.

On top of the point-to-point communication analysis, the cost of collective communication will increase with the concurrency. Studies on other architectures have shown that it is insignificant at 64 processors, but grows to 25% of the total communication time at 2048 processors.

It is not unreasonable to suggest that the improved efficiency in the computational portion of the optimized version of the code results in a greater percentage of time spent in communications. In the dual-core optimized-code case this effect and the increased costs of communication in dual core mode discussed previously work together, and a very large dual-core penalty is seen at high concurrencies.

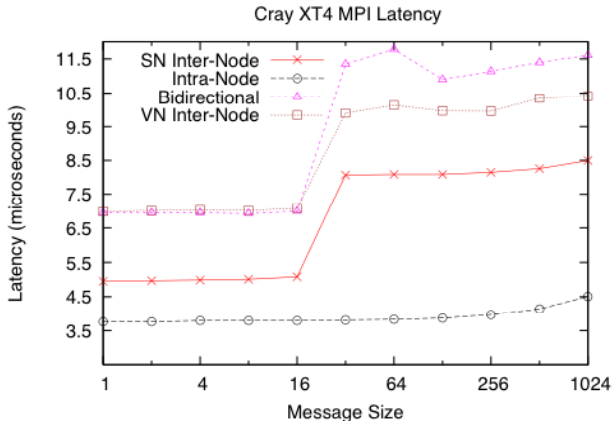

Figure 3.2 Observed Time for Small MPI Messages on Jaguar XT4.

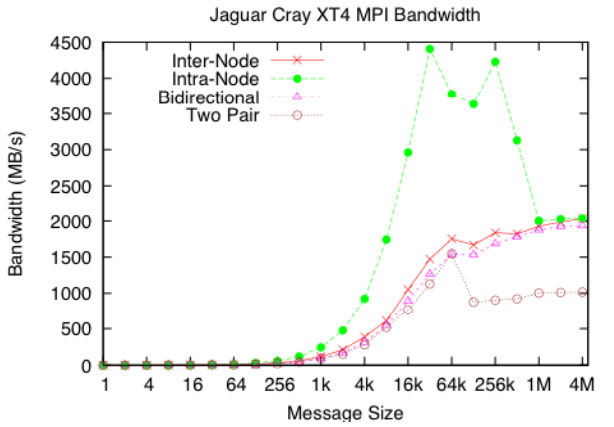

Figure 3.3 Observed MPI Bandwidth on Jaguar XT4.

4 BeamBeam3D

BeamBeam3D [7] models the colliding process of two counter-rotating charged particle beams moving at close to the speed of light. An accurate modeling of the beam-beam interaction

is essential to maximizing the luminosity in high energy accelerator ring colliders. The BeamBeam3D application performs a 3D particle-in-cell computation that contains multiple models (weak-strong, strong-strong) and multiple collision geometries (head-on, long-range, crossing angle). It tracks macroparticles in colliders using a transfer map. The simulated particles are deposited onto a three-dimensional grid to calculate the 3D charge density distribution. At collision points, the electric and magnetic field are calculated self consistently by solving the Vlasov-Poisson equation using Hockney's FFT method. Then the electric field and magnetic field are calculated on the grid and reinterpolated back to the macroparticles. The macroparticles are advanced in momentum space using these fields plus external fields from accelerator forces and focusing elements. The parallel implementation utilizes a particle-field decomposition method to achieve load balance. BeamBeam3D's communication is dominated by the expensive global operations to gather the charge density, broadcast the electric and magnetic fields, and perform transposes for the 3D FFTs—this represents a high volume of global message exchange communication.

For the strong-scaling experiments conducted in this study, we examine a 5 million particle simulation using grid resolutions of 256x256x32; comparative performance data are shown in Figure 4.1.

BeamBeam3D has some very complex communication issues, such as load-imbalance, that are not present in many other applications. The communication issues are less prevalent at the lower concurrencies. For example, with XT4 dual core, communication accounts for 4% of total time with 8 cores, and 15% with 64 cores. The bandwidth and latency effects discussed for MILC could be playing a larger role at higher concurrencies.

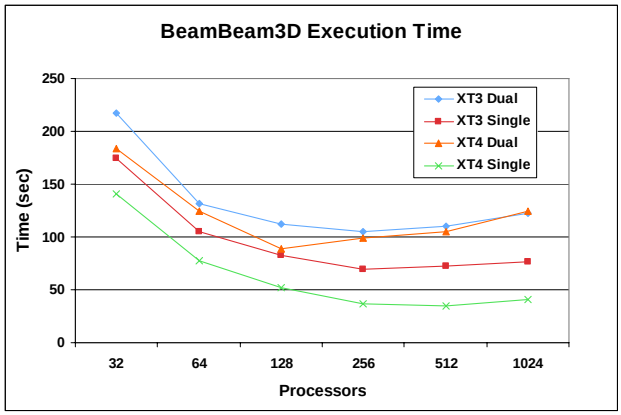

Figure 4.1 Performance of BeamBeam3D on Jaguar.

Timings for the 64-processor runs are shown in Table 4.1 below. We can see that the dual core penalty is significant on both architectures, but higher on the XT4.

	Times (seconds)	
Cores	**XT3**	**XT4**
Single Core	86	77
Dual Core	109	102
	Dual Core Penalty	
	XT3	**XT4**
	1.27	1.32

Table 4.1 BeamBeam3D Timings on Jaguar.

Interestingly, we see the best performance on 256 processors for both the XT3 single and dual core runs, but on the XT4 it is 128 processors for dual core and 512 processors for single core. We observe a large communication increase (22% with 128 cores *vs.* 43% with 256 cores) in XT4 dual-core mode. It seems that the XT4 node has a different balance with respect to the interconnect and computation in dual-core mode, and this has a major impact on scalability.

5 Predicting the Performance of Applications on Future Cray XT Systems

In a previous report [5], we describe in some detail the architectural similarities of the AMD Opteron single and dual core chips. We also

observe that when excluding messaging performance, the primary source of contention when moving from single core to dual core is memory bandwidth. Testing with STREAM and Membench micro-benchmarks confirms this assumption, as the performance of the 2.6 GHz AMD cores are nearly identical when the data fits in the L2 cache, and only becomes differentiated when the data sizes become larger than the L2 cache and must go to main memory. In addition, an analysis with Apex-MAP shows that a simple 2-parameter (bandwidth + latency) performance model that assumes dual-core effectively halves the main memory bandwidth, is highly accurate in predicting dual-core performance for a wide variety of memory access patterns. Therefore, investigation of more complex models for dual-core performance is unlikely to yield higher-fidelity results for this chip architecture.

With this in mind, we can attempt the task of extrapolating the performance of the applications studied here on quad-core systems. We begin by enumerating the assumptions of our model:

1. The only source of performance difference between single- and dual-core runs is memory bandwidth contention.
2. The 2.6 GHz RevE and RevF AMD cores execute code at roughly the same performance in the absence of memory bandwidth contention.
3. We can therefore break execution time into the portion that is stalled on shared resources (memory bandwidth) and the portion that is stalled on non-shared resources (everything else).
4. Under this circumstance, we can use the timing difference from single- to dual-core runs to compute the fraction of execution time spent in memory bandwidth contention.
5. We can then extrapolate the quad-core performance by assuming the time spent in the execution component remains the same, but the time spent in memory bandwidth contention will increase proportional to the reduction in effective memory bandwidth per core.

We note that the quad-core AMD architecture has some micro-architectural changes, such as the improved SIMD throughput, changes in cache architecture, and TLB, that are not accounted for in our model. The core improvements offer limited up-side potential in terms of performance of the quad-core. Our model represents the baseline case where **no** improvements to the core are assumed.

In addition to the previously described MILC and BeamBeam3D applications, we have collected compute-only timing information for three other scientific applications, CAM, GTC and PARATEC.

The Community Atmosphere Model (CAM) is the atmospheric component of the flagship Community Climate System Model (CCSM3.0). Developed at the National Center for Atmospheric Research (NCAR), the CCSM3.0 is used to study climate change. The CAM application is an atmospheric general circulation model (AGCM) and can be run either coupled within CCSM3.0 or in a stand-alone mode driven by prescribed ocean temperatures and sea ice coverages [8]. AGCMs are key tools for weather prediction and climate research. They also require large computing resources: even the largest current supercomputers cannot keep pace with the desired increases in the resolution and simulation times of these models. The version of CAM we used is a D resolution (about 0.5 degree resolution) with finite-volume dynamical core.

GTC is a 3-D particle-in-cell code used for studying turbulent transport in magnetic fusion plasmas [9]. The simulation geometry is that of a torus, which is the natural configuration of all tokamak fusion devices. As the charged particles forming the plasma move within the

externally-imposed magnetic field, they collectively create their own self-consistent electrostatic (and electromagnetic) field that quickly becomes turbulent under driving temperature and density gradients. Waves and particles interact self-consistently with each other, exchanging energy that grows or damps their motion or amplitude. The particle-in-cell (PIC) method describes this complex phenomenon by solving the 5D gyro-averaged kinetic equation coupled to the Poisson equation. The test case studied here is 10 particles per cell and 2000 time steps.

PARATEC (Parallel Total Energy Code) performs ab-initio quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set [10]. The pseudopotentials are of the standard norm-conserving variety. Forces and stress can be easily calculated and used to relax the atoms into their equilibrium positions. PARATEC uses an all-band conjugate gradient (CG) approach to solve the Kohn-Sham equations of Density Functional Theory (DFT) and obtain the ground-state electron wave functions. Much of the computation time (typically 60%) involves FFTs and BLAS3 routines. In solving the Kohn-Sham equations using a plane wave basis, part of the calculation is carried out in real space and the remainder in Fourier space using parallel 3D FFTs to transform the wave functions between the two spaces. The global data transposes within these FFT operations account for the bulk of PARATEC's communication overhead, and can quickly become the bottleneck at high concurrencies. The test case used as input to collect data is bulk silicon with a unit cell containing 125 atoms, running a single self-consistent field calculation.

The application timings on Jaguar XT3 and XT4 with single and dual cores are summarized in Table 5.1. All applications are run with 64 processors except for CAM which uses only 56 processors.

Application	Single Core		Dual Core		Dual Core Penalty	
	XT3	XT4	XT3	XT4	XT3	XT4
MILC	274	230	358	277	1.31	1.20
MILC-opt	160	127	230	181	1.44	1.43
BeamBeam3D	86	77	109	102	1.27	1.32
CAM	1123	1043	1283	1126	1.14	1.08
GTC	1389	1348	1447	1398	1.04	1.04
PARATEC	609	598	620	612	1.02	1.02

Table 5.1 Compute Only Times (seconds) for Applications on Jaguar.

We began by testing our model by using the XT3 performance data to predict the effective performance on the XT4. The XT4 in this test operates at the same clock frequency as the XT3, but the DDR2-667 memory subsystem is 30% faster than the DDR1-400 MHz memory of the XT3. Using the MILC optimized version data:

- The execution time for single-core runs on the XT3 is 160 seconds, and the time spent in dual-core is 230 seconds.

- The STREAM benchmarks indicate that the memory bandwidth for dual core is approximately half that of the single core (for this example we will assume that it is half), so if the five assumptions

above hold true, we should expect execution time to obey the relationship:
  - single core: core_exec_time + bandwidth_contention_time = 160 s
  - dual core: core_exec_time + 2*bandwidth_contention_time=230 s

- Solving the above system of equations provides us with an estimate of 90 seconds spent executing in the core (for both single and dual core) and 70 seconds spent in memory bandwidth contention for single core and 140 seconds (2x longer) spent in memory bandwidth contention for dual core.

We can now use the STREAM bandwidth numbers to project the time spent in memory bandwidth contention for the XT4's faster memory subsystem and predict single and dual core XT4 times. In Table 5.2 below we show predictions and actual measurements for the XT4 using STREAM TRIAD rates both in determining the memory bandwidth contention time, and in extrapolating to the XT4. The relatively large prediction errors for MILC, MILC-opt, and BeamBeam3D 64-core in dual core XT4 performance could be associated with communication effects that are not accounted for in the model. We note that the error in the prediction of 8-core Beam-Beam3D runs is considerably smaller than that of the 64-core runs that we present in Table 5.2. However, we also point out that overall, the predictive errors are surprisingly small given the simplicity of the model.

The predicted quad-core penalties for MILC, MILC-opt, and BeamBeam3D are relatively large. The AMD quad-core chip (Barcelona) offers some mitigating features in the core design, such as a doubling in floating point throughput, which could reduce this penalty, but the potential for performance improvements are bounded by the low Computational Intensity (CI) measured for these codes. The impact of the changes in cache hierarchy is not understood, but the proportion of total on-chip cache to the number of cores remains the same although the hierarchical arrangement of the memory in the cache has changed. So, although we ignore the micro-architectural changes, the results are not likely to change dramatically from this simple model.

Application	**XT4 Single Core**			**XT4 Dual Core**			**XT4 Quad Core**	
	Prediction	**Actual**	**Error**	**Prediction**	**Actual**	**Error**	**Prediction**	**Penalty**
MILC	227	230	-1.5%	289	277	4.3%	410	1.78
MILC-opt	120	127	-5.1%	172	181	-4.7%	273	2.15
BB3D,64 core	73	77	-5.2%	90	102	-11.7%	123	1.60
BB3D,8 core	494	516	-4.3%	595	600	-0.8%	793	1.54
CAM	1032	1043	-1.0%	1151	1126	2.3%	1382	1.33
GTC	1356	1348	0.6%	1399	1398	0.1%	1483	1.10
PARATEC	603	598	0.8%	611	612	-0.2%	627	1.05

Table 5.2 Prediction and Actual Compute-only Times (in seconds) for Applications.

6 Conclusions

We have examined in detail the performance of the MILC and BeamBeam3D applications on the Cray XT3 and XT4 in order to quantify the performance implications of running on a dual core configuration. In both cases a performance penalty of more than 20% is seen at low

concurrencies, and this increases with concurrency. In the case of MILC, optimizations to increase throughput of the memory subsystem are seen to be less effective on the dual core system, dealing the optimized code a double blow on dual-core systems.

In addition, we have developed a simple performance prediction model based on effective memory bandwidth and validated this model on the XT4 with three additional applications, CAM, GTC and PARATEC. The results are extremely promising, with all but one error rates of less than 10%. Our predictive model for a quad-core system are obviously subject to a much wider variety of errors, but nevertheless seem to show that for some applications the transition to quad-core will be a challenge for highly-optimized applications, such as MILC that are memory bandwidth bound, but on average will present more modest impacts on other mainstream applications such as CAM, GTC, and PARATEC.

Acknowledgments

The authors would like to thank Cray and AMD technical staff for useful discussions. This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. The authors are supported by the Director, Office of Science, Advanced Scientific Computing Research, U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] S. Borkar. “Design challenges of technology scaling.” *IEEE Micro*, 19(4):23–29, Jul-Aug 1999.

[2] P. P. Gelsinger. “Microprocessors for the new millennium: Challenges, opportunities, and new frontiers.” In *International Solid State Circuits Conference, ISSCC*, pages 22–25, 2001.

[3] J. L. Hennessy and D. A. Patterson. “*Computer Architecture : A Quantitative Approach; fourth edition*.” Morgan Kaufmann, San Francisco, 2006.

[4] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. Lester Plishker, J. Shalf, S. Webb Williams and K. A. Yelick, “The Landscape of Parallel Computing Research: A View from Berkeley,” *EECS Department University of California, Berkeley Technical Report No. UCB/EECS-2006-183* December 18, 2006.

[5] J. Levesque, J. Larkin, M. Foster, J. Glenski, G. Geissler, S. Whalen, B. Waldecker, J. Carter, D. Skinner, H. He, H. Wasserman, J. Shalf, H. Shan, E. Strohmaier. “Understanding and Mitigating Multicore Performance Issues on the AMD Opteron Architecture,” LBNL Report 62500, March 2007.

[6] MILC collaboration's public lattice gauge theory code. See http://physics.utah.edu/~detar/milc.html

[7] J. Qiang, M. Furman, and R. Ryne. A parallel particle-in cell model for beam-beam interactions in high energy ring colliders. J. Comp. Phys., 198, 2004.

[8] W. D. Collins, P. J. Rasch, B. A. Boville, J. J. Hack, J. R. McCaa, D. L. Williamson, B. P. Briegleb, C. M. Bitz, S.-J. Lin, and M. Zhang. The Formulation and Atmospheric Simulation of the Community Atmosphere Model: CAM3. *Journal of Climate*, 19, 2006.

[9] Z. Lin, T. S. Hahm, W. W. Lee, W. M. Tang, and R. B. White. Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science*, Sep 1998.

[10] PARAllel Total Energy Code Webpage. http://www.nersc.gov/projects/paratec.