Performance of the Community Atmosphere Model on the Cray X1E and XT3 *

P. H. Worley †

Abstract

The Community Atmosphere Model (CAM) is the atmospheric component of the Community Climate System Model (CCSM) and is the primary consumer of computer resources in typical CCSM simulations. We analyze the performance of CAM on the Cray XT3 and Cray X1E systems at Oak Ridge National Laboratory, describing the performance sensitivities of the two systems to the numerous tuning options available in CAM. We then compare performance on the Cray systems with that on a number of other systems, including the Earth Simulator, an IBM POWER4 cluster, and an IBM POWER5 cluster.

1 Introduction

Oak Ridge National Laboratory (ORNL) is the site of large configurations of two Cray high performance computing products, a 1024 processor Cray X1E and a 5294 processor Cray XT3. These systems are part of the National Center for Computational Sciences (NCCS) [14], and are dedicated to a limited number of high-impact, grand challenge scale computational science projects. Among the projects currently awarded time is the Climate-Science Computational End Station Development and Grand Challenge Team (CCES). The CCES is concerned with advancing climate science through climate model development and climate model simulations. Both development and simulation activities utilize the Community Climate System Model (CCSM).

The CCSM is a fully-coupled, global climate model that provides state-of-the-art computer simulations of the Earth's past, present, and future climate states, and is an important tool in understanding climate change [2, 5]. The CCSM is made up of four component models (atmosphere, ocean, land, and sea ice) and a coupler. The Community Atmosphere Model (CAM) is the atmospheric component of the CCSM and is the primary consumer of computer resources in typical CCSM simulations. As such, CAM performance evaluation and optimization on the NCCS systems is an important aspect of CCES model development activity.

CAM performance has been documented in a number of recent papers [1, 13, 16, 21, 22]. This paper provides a more detailed description of CAM performance on the X1E and the XT3 than is contained in these other papers, and focuses on using CAM to illuminate performance characteristics of these two systems.

The outline of the paper is as follows. Section 2 is a brief description of CAM. Section 3 is a list of the platforms mentioned in the paper. Section 4 begins with a brief description of the X1E, followed by a discussion of issues in CAM porting and performance tuning on the X1E. Section 5 contains a similar discussion for the XT3. Section 6 is a comparison of the performance of CAM on the X1E and XT3 with that on a number of other high performance computing platforms.

2 Community Atmosphere Model

CAM is developed at the National Science Foundation's National Center for Atmospheric Research (NCAR) with contributions from researchers funded by the Department of Energy and by the National

*This research was sponsored by the Climate Change Research Division of the Office of Biological and Environmental Research and by the Office of Mathematical, Information, and Computational Sciences, both in the Office of Office of Science, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Batelle, LLC. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

†Computer Science and Mathematics Division, Oak Ridge National Laboratory, P.O. Box 2008, Bldg. 5600, Oak Ridge, TN 37831-6016 (worleyph@ornl.gov)

Aeronautics and Space Administration [3, 4]. CAM is a mixed-mode parallel application code, using both the Message Passing Interface (MPI) [8] and OpenMP protocols [6]. CAM is characterized by two computational phases: the dynamics, which advances the evolution equations for the atmospheric flow, and the physics, which approximates subgrid phenomena such as precipitation processes, clouds, long- and short-wave radiation, and turbulent mixing [4]. The approximations in the physics are referred to as the physical parameterizations. The physics phase also includes calls to land, sea ice, and ocean models. In the CAM experiments described in these papers, the sea ice and ocean models are simplified data models, not the active models typically used in CCSM simulations. However, the land model is the Community Land Model (CLM) [15], the same as used in CCSM. Control moves between the dynamics and the physics at least once during each model simulation timestep.

CAM includes multiple options for the dynamics, referred to as *dynamical cores* or *dycores*, one of which is selected at compile-time. Three dycores are currently supported: a spectral Eulerian (EUL) [9], a spectral semi-Lagrangian (SLD) [20], and a finite volume semi-Lagrangian (FV) [10]. The spectral and finite volume dycores use different computational grids. An explicit interface exists between the dynamics and the physics, and the physics data structures and parallelization strategies are independent from those in the dynamics. A dynamics-physics coupler moves data between data structures representing the dynamics state and the physics state.

As a community model, it is important that CAM run efficiently on different architectures, and that it be easily ported to and optimized on new platforms. In support of this, CAM contains a number of compile-time and runtime parameters that can be used to optimize performance for a given platform, problem or processor count [21]. Those that were important when porting and tuning CAM performance on the X1E and XT3 are described below.

2.1 Platform-specific Code

CAM has a number of code fragments, delimited by C preprocessor (`cpp`) directives, that are enabled only for certain systems. For example, there are a few routines for which we were unable to develop a single version that runs well on both vector and nonvector systems. `cpp` is used to choose either the vectorizable or the cache-friendly versions of these routines. `cpp` is also used to choose between math library routines with different calling sequences, FFT routines primarily.

2.2 Physics Data Structures

The physics uses the same computational grid as the dynamics. All three dycores employ a tensor product longitude-latitude-vertical ($\texttt{nlon} \times \texttt{nlat} \times \texttt{nver}$) grid covering the sphere. We refer to all grid points in this three-dimensional grid with a given horizontal location, differing only in the vertical coordinate, as a *vertical column*, or just *column*. The current physical parameterizations in CAM are based on vertical columns, and physics computations at a given timestep are independent between columns. The basic data structure in the physics is the *chunk*, an arbitrary collection of vertical columns. Grid points in a chunk are referenced by (`local column index`, `vertical index`). A "chunked" array is declared as (`pcols`, `nver`, `nchunks`) and the loop structure is

```
do j=1,nchunks
  do k=1,nver
    do i=1,ncols(j)
      (physical parameterizations)
    enddo
  enddo
enddo
```

Here

- `ncols`(j) is the number of columns allocated to chunk j;
- `nchunks` is the number of chunks;
- `pcols` is the maximum number of columns allocated to any chunk (specified at compile time).

Thus, $\texttt{pcols} \cdot \texttt{nchunks} \geq \texttt{nlon} \cdot \texttt{nlat}$ for a tensor-product longitude-latitude grid, but there are no other assumptions about the composition of a chunk. In particular, the columns bundled in a given chunk need not be geographically contiguous.

Physics performance tuning options include the following.

1. The compile-time parameter `pcols` determines the maximum number of columns assigned to a chunk. (Depending on the number of processors and number of columns, `ncols(j)` can be less than `pcols`.) Large `pcols` generates long inner loops, which improve vectorization. Small `pcols` decreases the size of the basic

computational unit, which improves cache locality. The specific value of `pcols` also determines the memory alignment of elements in the chunked arrays, which has performance implications on most systems.

2. The time required to process a column is a function of geographical location and simulation time, and excellent static load balancing schemes are known. However, the load balancing schemes are at odds with the domain decompositions utilized by the dycores, thus requiring interprocess communication to implement. Four load balancing options, representing different ratios of interprocess communication overhead and load imbalance, are supported. We refer to the best load balancing scheme, which incurs the highest communication overhead, as "full" load balancing.

3. The communication protocol used to implement the interprocess communication required by the load balancing scheme is a runtime option. Possibilities include MPI collectives, MPI two-sided point-to-point implementations, MPI one-sided point-to-point implementations, and Co-Array Fortran one-sided point-to-point implementations.

2.3 Spectral Dynamics

The spectral dycores support only a one-dimensional decomposition of the computational grid, over latitude initially. Each call of the spectral dynamics moves back and forth between the longitude-latitude-vertical grid point space and the spectral coefficient space. The parallel algorithm implementing these transforms changes the decomposition from one-dimensional over latitude to one-dimensional over longitude and back again. The communication protocols for these interprocess communications is a runtime option. The options are the same as for the physics load balancing, but different choices can be made for the physics and for the dynamics.

2.4 Finite Volume Dynamics

The finite volume dycore employs a two-dimensional block decomposition of the computational grid. There are two computational phases in this dycore. In one, the longitude and latitude dimensions are decomposed. In the other, the latitude and vertical dimensions are decomposed. Interprocess communication is needed for remapping between the two decompositions.

The FV dycore tuning options include the following.

1. The specification of the virtual two-dimensional processor grid that defines the blocks for the latitude-vertical domain decomposition and the specification of the virtual two-dimensional processor grid that defines the blocks for the longitude-latitude domain decomposition are runtime options. Note that a one-dimensional decomposition over latitude is best for small process counts as it eliminates the need for a remap.

2. The communication protocol used for the remap is a runtime option. Options include both MPI two-sided and one-sided point-to-point implementations, using either temporary contiguous send and receive buffers or sending from and receiving into MPI derived types.

2.5 Parallel Programming Paradigm

CAM supports MPI, OpenMP, and hybrid MPI/OpenMP parallelism. OpenMP is not available on the XT3 currently, but is an option on the X1E (and many other target architectures). The same number of MPI processes is required in the dynamics and the physics, but it is possible to use more OpenMP threads in one part of the code than in another. For example, the physics has considerably more "exposed" parallelism than the dynamics. The primary advantage of OpenMP is in just this situation, allowing more processors to be applied to the physics when parallelism is exhausted in the dynamics. OpenMP parallelism also allows the number of MPI processes to be decreased for a fixed number of processors, which can decrease MPI communication overhead.

3 Experimental Platforms

For comparison purposes, performance data were collected on a number of different high performance computing systems, as described below. Brief descriptions of the Cray X1E and XT3 are included in this list, but expanded descriptions of these architectures are provided in Sections 4 and 5, respectively.

- Cray X1 at ORNL: 512 Multi-Streaming processors (MSP), each capable of 12.8 GFlop/s for 64-bit operations. MSPs are fully connected within 16-MSP subsets, and are connected via a 2-D torus between subsets.

- Cray X1E at ORNL: 1024 MSPs, each capable of 18 GFlop/s for 64-bit operations. MSPs are fully connected within 32-MSP subsets, and are connected via a 2-D torus between subsets. This system is an upgrade of the original Cray X1 at ORNL.
- Cray XT3 at ORNL: 5294 single processor nodes (2.4 GHz AMD Opteron) and a 3-D torus interconnect. Each processor is capable of 4.8 GFlop/s for 64-bit operations.
- Earth Simulator: 640 8-way symmetric multiprocessor (SMP) nodes and a 640x640 single-stage crossbar interconnect. Each processor has 8 64-bit floating point vector units running at 500 MHz, and is capable of 8 GFlop/s for 64-bit operations.
- IBM p575 cluster at the National Energy Research Supercomputer Center (NERSC): 122 8-way p575 SMP nodes (1.9 GHz POWER5 processors) and an HPS interconnect with 1 two-link adapter per node. Each processor is capable of 7.6 GFlop/s for 64-bit operations.
- IBM p690 cluster at ORNL: 27 32-way p690 SMP nodes (1.3 GHz POWER4 processors) and an HPS interconnect. Each node has two HPS adapters, each with two ports. Each processor is capable of 5.2 GFlop/s for 64-bit operations.
- IBM SP at NERSC: 184 Nighthawk II 16-way SMP nodes (375 MHz POWER3-II processors) and an SP Switch2. Each node has two interconnect interfaces. Each processor is capable of 1.5 GFlop/s for 64-bit operations.
- Itanium2 cluster at Lawrence Livermore National Laboratory: 1024 4-way Tiger4 nodes (1.4 GHz Intel Itanium 2) and a Quadrics QsNetII Elan4 interconnect. Each processor is capable of 5.6 GFlop/s for 64-bit operations.
- SGI Altix 3700 at ORNL: 256 1.5 GHz Itanium2 processors and a NUMAlink switch. The machine has an aggregate of 2 TByte of shared memory. Each processor is capable of 6.0 GFlop/s for 64-bit operations.

4 Cray X1E

4.1 X1E Overview

The X1E is Cray's scalable vector architecture. The X1E is characterized by high-speed custom vector processors, high memory bandwidth, and a high-bandwidth, low-latency interconnect. The logical view of an X1E is as a cluster of 4-way SMP SMP nodes made up of four vector processors where each vector processor has eight vector units.

The architecture is more complicated than this simple view would imply. The vector processor, or Multi-Streaming Processor (MSP), is itself made up of four Single Streaming Processors (SSP). Each SSP has two 32-stage 64-bit wide vector units running at 1.13 GHz and one 2-way superscalar unit running at 565 MHz. The four SSPs share a 2MB cache. While the user has the option to treat each SSP as a processor, the compiler often does an excellent job assigning work to all vector units, allowing the user to consider each MSP a single vector processor with a peak performance of 18 GFlop/s (64 bit) instead of four SSPs each with a performance of 4.5 GFlop/s.

Two MSPs are implemented in a single multichip module (MCM). However these two MSPs are not in the same SMP node. Instead four MCMs together make up 2 SMP nodes. These 8 MSPs, called a compute module, share bandwidth into the interconnect. The interconnect is fully connected within subsets of 4 compute modules, or 32 MSPs. The interconnect is either a hypercube or a two-dimensional torus between these 32-MSP subsets.

Each MSP can read 33.5 GByte/s from main memory or the interconnect and can write between 12.8 and 20.5 GByte/s. The interconnect supports 51.2 GByte/s each direction from/to a compute module.

The X1E at ORNL has 256 SMP nodes (1024 MSPs) and 8 GBytes of memory per node. The interconnect is configured as a two-dimensional torus between the 32-MSP fully connected subsets.

4.2 Porting and Tuning

CAM was ported to the Cray X1, the predecessor to the X1E, previously [7]. The X1 port of CAM worked on the X1E without change. The X1E differs from the X1 in that it has 41% higher peak computation rate per MSP, lower memory latency, increased contention for memory bandwidth and increased contention for interconnect bandwidth. These differences between the X1 and X1E are reflected in the CAM performance characteristics. For example, Figure 4.1 is a plot of the runtime of the physics, normalized by the fastest runtime, as a function of `pcols`. On both systems the optimal `pcols` is approximately 514, and performance degrades when `pcols` is a power-of-two. However, the performance

sensitivity to `pcols` is smaller on the X1E. This is probably due to the increased memory contention on the X1E offsetting some of the advantage of large `pcols` on vectorization.

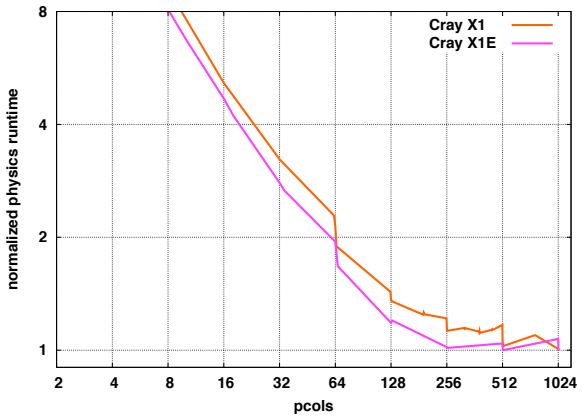

FIGURE 4.1: X1E `pcols` Performance Sensitivity

The differences in the performance characteristics were not large enough to change the optimal performance tuning settings.

- Long inner loops (large `pcols`) are better, to a point.
- Eliminating physics load imbalances is more important than decreasing interprocess communication overhead.
- MPI collectives are faster than point-to-point implementations (when using "full" load balancing), even compared to using Co-Array Fortran in the point-to-point implementation.
- Do not use MPI derived types when using point-to-point implementations.
- Pure MPI is faster than hybrid MPI/OpenMP for almost all processor counts (for CAM).
- Use MSPs, not SSPs, i.e., assign one process or thread per MSP and let the compiler assign work to functional units within the MSP.

We henceforth use "processor" and "MSP" interchangeably when discussing the X1 and X1E.

While the X1 port worked well, CAM is still evolving, and most of the new code is not developed on the X1E. Figure 4.2 contains graphs of the performance of CAM as a function of version number. The versions named on the X-axis were developed on the X1E, and often include changes that eliminate performance degradations that crept in since the previous X1E-oriented modification. For each named version, we also measured performance for the immediately preceding version. The name of each version is of the form "3.X_Y". The "3." is dropped from the name in the graph where it improves readability.

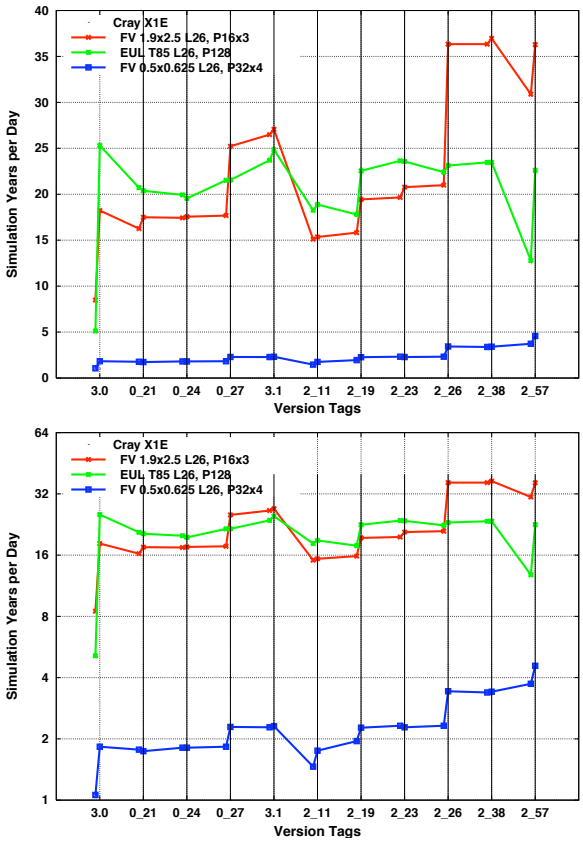

FIGURE 4.2: Performance History on the X1E

The two graphs are identical except that the lower graph uses a logarithmic scaling of the Y-axis. Three benchmark problems were used in these experiments.

1. EUL dycore running on 128x256x26 (latitude by longitude by vertical) computational grid. This dycore and grid, referred to as T85 L26, are used in production CCSM simulations currently.
2. FV dycore running on 96x144x26 computational grid. This grid is referred to as 1.9x2.5 L26. The FV dycore is the preferred dycore for atmospheric chemistry due to its conservation properties. 1.9x2.5 L26 is the proposed grid for production CCSM simulations using FV.

3. FV dycore running on 361x576x26 computation grid. This grid is referred to as 0.5x0.625 L26. This problem size is 15 times larger than 1.9x2.5 L26 and is considered a very high, but feasible, resolution for climate simulations.

For T85 L26 we used 128 processors, the maximum possible without using OpenMP parallelism. For 1.9x2.5 L26, we used 48 processors in a 16x3 two-dimensional virtual processor grid. For 0.5x0.625 L26, we used 128 processors in a 32x4 two-dimensional virtual processor grid. In all cases we set `pcols` to 514 and ran with internal performance timers enabled. These choices degrade performance somewhat, as shown in Section 6, but the graphs still document the qualitative nature of the performance history on the X1E. It is clear from these results that maintaining performance as CAM evolves is as important as pursuing further performance improvements.

5 Cray XT3

5.1 XT3 Overview

The XT3 is Cray's third-generation massively parallel processing system. It follows a similar design to the successful Cray T3D and Cray T3E [19] systems. As in these previous systems, the XT3 builds upon a single processor node. The XT3 connects these nodes with a customized interconnect managed by a Cray-designed application-specific integrated circuit called SeaStar.

Each XT3 node has one Opteron processor with its own dedicated memory and communication resources. The XT3 has two types of nodes: compute and service. The compute nodes are optimized for application performance and run a lightweight operating system kernel called Catamount. The service nodes run SuSE Linux and are configured for I/O, login, network, or system functions.

The XT3 at ORNL uses Opteron model 150 processors clocked at 2.4 GHz. This model includes an Opteron core, integrated memory controller, three 16bit-wide 800 MHz HyperTransport (HT) links, and L1 and L2 caches. The Opteron core has three integer units and one floating point unit capable of two floating-point operations per cycle, achieving a peak floating point computation rate of 4.8 GFlop/s (64 bit). Each compute node has 2 GBytes of memory. Latency to main memory is 50-60 nsec, and peak bandwith to main memory is 6.4 GByte/s.

Each XT3 node is connected to the Cray SeaStar chip via a 6.4 GByte/s HT link. The SeaStar chip provides six network links to connect to neighbors in a three-dimensional torus/mesh topology. Each of the six links has a peak bandwidth of 7.6 GByte/s with sustained bandwidth of around 4 GByte/s.

The XT3 at ORNL has 5212 compute nodes and 82 service nodes. The nodes are connected in a three-dimensional mesh of size 14 x 16 x 24, with torus links in the first and third dimension. For the experiments described in this paper we used Fortran and C compilers from The Portland Group.

For more details on configuration and performance of the XT3 at ORNL, see Alam, et al [1].

5.2 Porting and Tuning

CAM had not previously been ported to an XT3, though there was prior experience running CAM on Opteron processors. A number of minor changes were required for the port.

1. Four routines were compiled with `-O1` compiler optimization, instead of the `-fast` optimization that was used with the rest of the code. This corrected runtime errors that occurred otherwise. Fortunately a lower level of compiler optimization for these routines does not degrade CAM performance.

2. The Catamount OS does not support a number of standard Linux OS system services. `cpp` directives were used to eliminate "optional" calls to unsupported system routines when running on the XT3.

3. Calls to the system routine `gettimeofday`, which were not optional, were replaced by calls to `MPI_Wtime`. This has since been adopted as the standard on all platforms when running with MPI.

4. Performance of writing to standard out and standard error was so poor that it qualified as a porting issue. The solution was to add a call to the system command `setvbuf` to enable full buffering and to allocate (65KB) buffers.

5. Execution of the code failed when reading/writing files in the NFS-attached file system. All experiments were run with all input and output files in the Lustre [11] parallel file system.

6. The MPI "eager" protocol caused system buffer space to be exhausted from the reception of unexpected messages when gathering data to a single process prior to writing output.

There are many possible solutions to this issue [17]. At the current time we have replaced the call to `MPI_Allgatherv` where the problem occurs with an MPI point-to-point implementation that limits the number of simultaneous sends to the root process. This solution works no matter how many processes are involved and does not impact performance of MPI communication in the rest of the code.

The next step after porting was to determine the optimal settings for the performance options. Figure 5.1 is a plot of normalized physics runtime as a function of `pcols`. Data are included for a number of other nonvector systems. Each system has somewhat different sensitivities and different optimal `pcols` values: 8 for the Altix; 24 for the p690; 34 for the XT3; 80 for the p575. However, they all are reasonably insensitive to the exact value as long as `pcols` is within a system-specific range, and neither very small (less than 8) nor very large values are efficient. (The p575 performance curve begins to increase more rapidly for `pcols` greater than 128.) While not shown here, the optimal `pcols` value on the XT3 is also sensitive to the level of compiler optimization. When using `-fastsse` the optimal `pcols` value is 40. We began using this `pcols` value in subsequent benchmarking runs, and continued to do so even after discovering the (slight) advantage of using a `pcols` value of 32. Note that compiling with `-fastsse` caused numerical problems and did not improve CAM performance appreciably over using `-fast`.

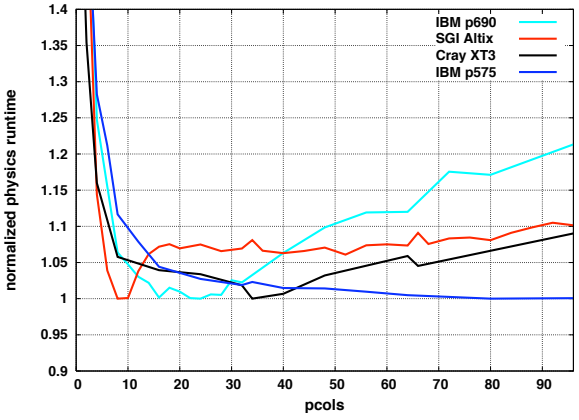

FIGURE 5.1: XT3 `pcols` Performance Sensitivity

A selection of optimal performance settings for the XT3 are as follows.

- `pcols` values between 34 and 40 are optimal, but values between 8 and 66 achieve acceptable performance.
- Eliminating physics load imbalances is more important than decreasing interprocess communication overhead. However, the preference is not as strong as on the X1E, and less than full load balancing is competitive in many instances.
- MPI collectives are faster than point-to-point implementations (when using full load balancing).
- Do not use MPI derived types when using point-to-point implementations. The disadvantage of using MPI derived types, while measurable, is not significant in most exeriments.
- I/O performance should be optimized wherever and however possible.

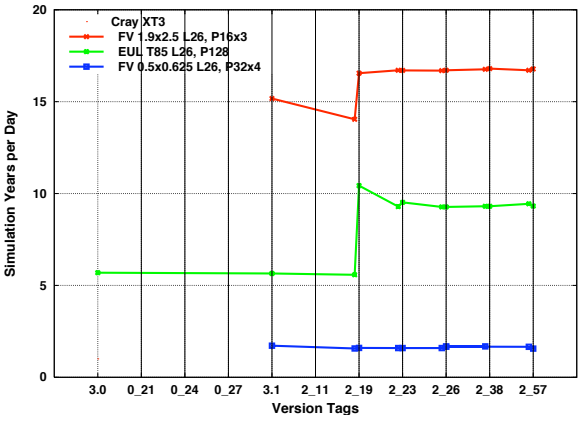

FIGURE 5.2: Performance History on the XT3

Figure 5.2 is analogous to Figure 4.2, graphing the performance of CAM on the XT3 as a function of version number. As can be seen, little system-specific code optimization has taken place on the XT3 beyond that which occurred in version 3.2_19. (The modifications required to port to the XT3 were introduced in version 3.1. These same modifications were backported to version 3.0p1 for the EUL dycore.) All experiments were run with `pcols` set to 40 and with internal performance timers enabled. These choices did not impact performance negatively. Optimal settings were used for the other performance options and all I/O was limited to files in the Lustre file system. Version 3.2_19 contains the modification in which buffering for standard output and standard error was added. The cost for this I/O is relatively fixed for the benchmarks, so affects the smaller benchmarks more strongly. Much of the I/O cost occurs within the physics, which is called more

frequently when using the spectral dycores, making the T85 benchmark results most sensitive to this optimization.

6 Performance Results

In this section we describe and analyze processor scaling for two of the benchmark problems: T85 L26 and 0.5x0.625 L26. Each data point represents the best performance observed on a given platform for a given processor count after optimizing over the CAM performance tuning options.

6.1 T85 L26

For benchmarking T85 L26 we took version 3.0 and backported relevant performance optimizations from more recent versions of the CAM, e.g., enabling buffering of output to standard error and standard out on the XT3. Figure 6.1 is a graph of CAM performance for the T85 L26 benchmark on the Cray X1, X1E, and XT3, the IBM p575 cluster, and the IBM p690 cluster. As indicated earlier, the EUL dycore limits the number of MPI processes to be no more than the number of latitudes, 128 processes for this benchmark. There is also a computational inefficiency introduced in the dynamics when fewer than two latitudes are assigned to an MPI process. Thus inefficient scaling when using more than 64 MPI processes is not just due to increased interprocess communication overhead. OpenMP parallelism is used on the IBM systems to avoid this inefficiency and to exploit more than 128 processors.

For the X1 and X1E experiments the internal performance timers were disabled and `pcols` was set to 1026 for 8, 16 and 32 processors, to 514 for 64 and 96 processors, and to 258 for 128 processors. The earlier experiments indicated that `pcols` settings of 258, 514, and 1026 achieved comparable performance. These particular choices take into account the granularity for a given processor count, optimizing the memory usage. On the XT3 and the p575 cluster `pcols` was set to 40 and 80, respectively, for all processor counts. On the p690 cluster optimal `pcols` values varied between 16, 24, and 32 as the processor count varied, but performance differences between the three choices were small. Full load balancing was used on all systems except the p690 cluster. On the p690 cluster the optimal load balancing setting varied with processor count, reflecting the relatively high cost of interprocess communication on that system.

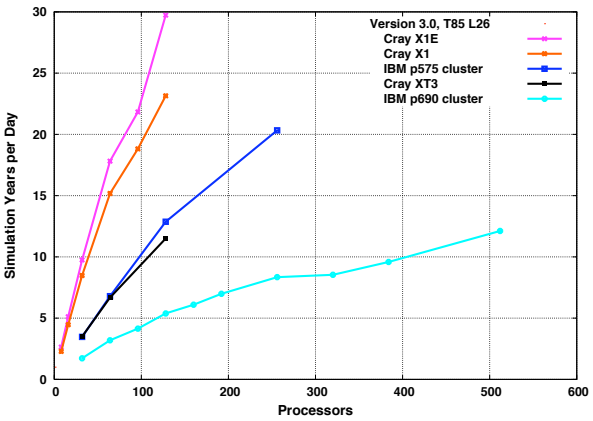

FIGURE 6.1: T85 L26 Performance

The X1E is 16% faster than the X1 for 8 through 96 processors. For 128 processors the X1E advantage jumps to 28%. The X1E is 2.3 to 2.8 times faster than the p575 cluster, 2.5 to 2.8 times faster than the XT3, and 5.5 times faster than the p690 cluster when using up to 128 processors. Using more processors on the X1E, via OpenMP, would decrease vector length in the physics below 256 for this benchmark, resulting in degraded single processor performance. In consequence, we would not expect performance to improve significantly beyond that achieved for 128 processes. In contrast, the IBM systems continue to scale to much higher processor counts, though they do not surpass the peak X1E performance with the available processor counts.

Performance on the XT3 is very similar to that on the p575 cluster for 32 and 64 processors. The p575 cluster performance is 12% faster for 128 processors. For 128 processors the p575 cluster is using 64 MPI processes with 2 OpenMP threads per process while the XT3 is using 128 MPI processes. Optimal p575 cluster results for 32 and 64 processors do not use OpenMP parallelism.

Figure 6.2 is a graph of seconds per simulation day for just the dynamics. From this it is clear that dynamics on the p575 cluster makes little use of more than 128 processors, and that dynamics on the p575 cluster is always faster than the dynamics on the XT3: by 35% for 32 processors, 46% for 64 processors, and 64% for 128 processors. Dynamics on the X1E is 2.8 times faster than that on the p575 cluster for 32 processors, 2.2 times faster for 64 processors, and 1.8 times faster for 128 processors. Examination of profile data indicates that the p575 cluster advantage over the XT3 occurs primarily in the computational phases for 32 and 64 processors. For 128 processors, exploitation of OpenMP parallelism also avoids the computational inefficiency caused by us-

ing more than 64 MPI processes and decreases the communication overhead relative to that seen on the XT3. The performance difference in computational phases would appear to be attributable to memory access patterns and instruction mix that allow the POWER5 processor to achieve processor efficiences comparable to those achieved on the Opteron. (Remember that the 1.9 GHz POWER5 has a 60% higher peak floating point operation rate than the 2.4 GHz Opteron.) The increasing competitiveness of the p575 cluster with respect to the X1E as a function of processor count is likewise due to the increasing importance of interprocess communication and the computational inefficiency caused by using more than 64 MPI processes.

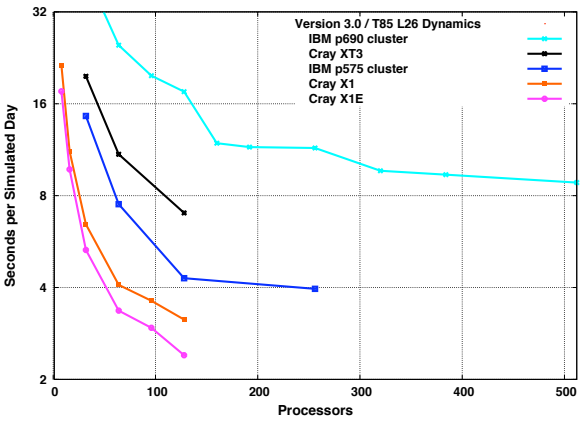

FIGURE 6.2: T85 L26 Dynamics

Figure 6.3 is a graph of seconds per simulation day for just the physics. The improvement in physics scalability from using OpenMP is clear. In contrast to the dynamics results, physics performance on the XT3 is better than that on the p575 cluster for the same number of processors: by 13% for 32 and 64 processors and by 6% for 128 processors. Physics performance on the X1E is between 2.6 and 2.8 times faster than on the p575 cluster up to 128 processors. Examining the performance profile indicates that the XT3 performance is similar or superior to that of the p575 cluster for most computational phases. This indicates a significantly different memory access pattern or instruction mix from the dynamics, one that favors the Opteron processor over the POWER5. For example, the physics is known to include a relatively large number of calls to the `sqrt` function and other intrinsics, which is not true of the dynamics.[1] The decreasing XT3 performance advantage over the p575 cluster is due to the increasing percentage of time spent in the land model and in a global sum as the processor count increases, both of which are faster on the p575 cluster than on the XT3.

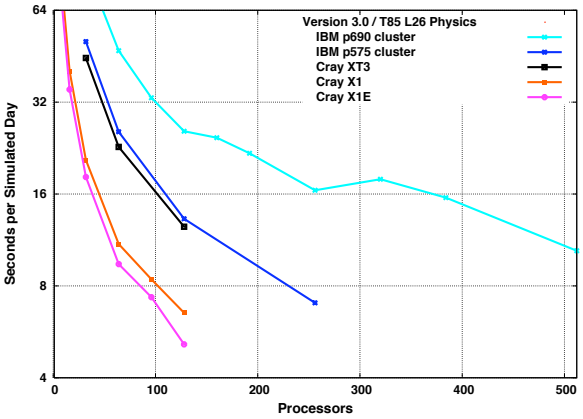

FIGURE 6.3: T85 L26 Physics

Figure 6.4 is a graph of seconds per simulation day for both dynamics and physics for the Cray X1E, Cray XT3, and IBM p575 cluster. While these data also appear in Figures 6.2 and 6.3, this graph shows clearly that the physics is between two and four times as expensive as the dynamics when not using more than 128 processors.

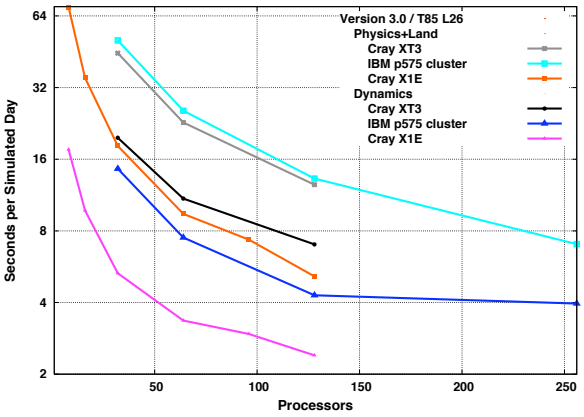

FIGURE 6.4: T85 L26 Diagnostics

[1]Note that both the MASS [12] and MASSV libraries were linked to for experiments on the p575 cluster, but no explicit attempt was made to use the vector MASS routines. Also, `-qstrict` was used to restrict compiler optimization on the IBM systems, which probably prevented the compiler from exploiting vector versions of the MASS replacement routines for the math intrinsic functions. CAM does not pass the numerical validation tests on the IBM systems without specifying `-qstrict` when using `-O3` and higher levels of compiler optimization.

6.2 0.5x0.625 L26

For the 0.5x0.625 L26 benchmark we took version 3.1 and, again, backported a few performance optimizations from more recent versions of the CAM. Figure 6.5 is a graph of CAM performance for this benchmark on the Cray X1, X1E, and XT3, the Earth Simulator, the IBM p575 cluster, the IBM p690 cluster, the IBM SP, and the Itanium2 cluster. The FV dycore requires that at least 3 latitudes and 3 vertical levels be assigned to each MPI process. For the 0.5x0.625 benchmark, the maximum two-dimensional virtual processor grid is then 120x8, implying a maximum of 960 MPI processes. The Earth Simulator and the IBM systems also use OpenMP parallelism, and can use more than 960 processors.

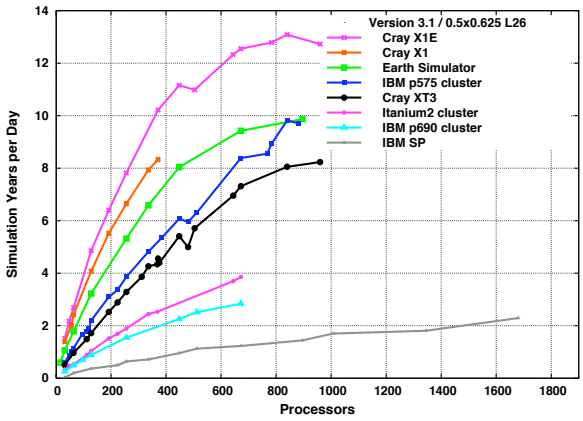

FIGURE 6.5: 0.5x0.625 L26 Performance

Unlike the T85 L26 benchmark experiments, internal performance timers were not disabled for the X1 and X1E experiments. Between version 3.0_27 and 3.1 the timing logic was modified and enabling the internal performance timers in version 3.1 does not have a significant performance impact on the Cray vector systems. For X1 and X1E experiments `pcols` was set to 870 for 32 to 256 processors, to 570 for 371 to 504 processors, to 330 for 644 to 784 processors, and to 258 for larger processor counts. On the IBM SP, the Itanium2 cluster, the XT3, the IBM p575 cluster, and the Earth Simulator `pcols` was set to 16, 16, 40, 80, and 512, respectively, for all processor counts. On the p690 cluster the optimal `pcols` value was 16 up to 128 processors, 24 for 224 and 256 processors, and 32 for higher processor counts.

Full load balancing was used on the X1, X1E, XT3, Earth Simulator, p575 cluster and p690 cluster. No load balancing was used on the IBM SP or on the Itanium2 cluster. One-dimensional domain decompositions were usually best until the number of MPI processes exceeded 64. Three types of two-dimensional decompositions were examined, defined by the following virtual processor grids: (1) $(P/4) \times 4$, (2) $(P/7) \times 7$, and (3) $(P/8) \times 8$, where P is the total number of MPI processes. Performance was typically best for the first two-dimensional decomposition until $(P/4) > 64$. Note that a two-dimensional decomposition was never optimal on the p690 cluster. Instead OpenMP parallelism was used to exploit additional processors.

The X1E is only 9% faster than the X1 for 32 processors, but the advantage grows to 19% for 128 processors, and reaches 22% for 371 processors. The Earth Simulator shows a similar scaling curve to the X1E, but the X1E is 1.5 times faster up to 256 processors, and retains a 1.3 times performance advantage out to 672 processors. The X1E is approximately 2.5 times faster than the p575 cluster for 32 processors, declining to 2 times faster for 256 processors, finally dropping to 1.3 times faster for 840 processors. The performance comparison with the XT3 is similar: 3 times faster for 32 processors, declining to 1.5 times faster for 960 processors. Performance on the XT3 is again most similar to that on the p575 cluster, though the XT3 performance lags behind that of the p575 cluster by between 12% and 22%. The p575 cluster uses OpenMP parallelism for many processor counts, and always uses OpenMP when using more than 336 processors.

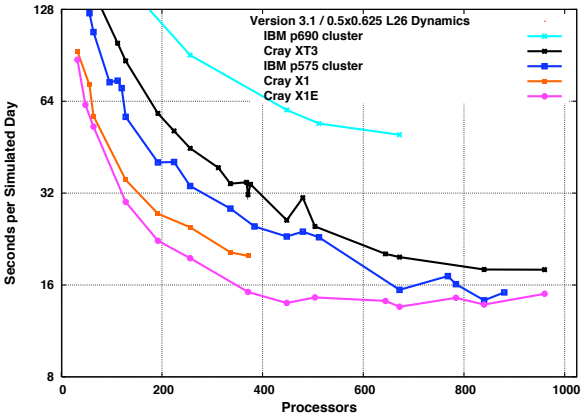

FIGURE 6.6: 0.5x0.625 L26 Dynamics

Figure 6.6 is a graph of seconds per simulation day for just the dynamics for five of the experimental platforms. Dynamics performance on the X1E is essentially flat when using more than 448 processors. The XT3 continues to see performance improvement out to 960 processors, though the improvement is marginal beyond 840 processors. It is difficult to

identify a trend in the p575 cluster results. Dynamics on the X1E is 3.2 times faster than on the XT3 for 32 processors, 1.9 times faster for 448 processors, and only 30% faster for 840 processors. Dynamics on the X1E is 2.3 times faster than on the p575 cluster for 32 processors, 1.7 times faster for 448 processors, and only 3% faster for 840 processors. Dynamics on the p575 cluster is faster than on the XT3 by 41% for 32 processors, by 13% for 448 processors, and by 26% for 840 processors. Note that the erratic behavior in the performance curves is a function of the two-dimensional virtual processor grid that is compatible with a given processor count. For example, on the XT3 we used a 64x7 virtual processor grid for 448 processors, a 120x4 grid for 480 processors, and a 72x7 grid for 504 processors. Decomposing latitude with 120 processes incurs a much higher communication overhead in the dynamics than decomposing using 64 or 72 processes.

From the profile data, the dynamics performance on the X1E suffers from both communication overhead and a load imbalance for large processor counts. The FV dycore applies a Fast Fourier Transform (FFT)-based filter to latitudes near the poles. As only a subset of the processes are assigned polar latitudes, the other processes are idle. The X1E also does not achieve good performance when calculating the FFTs. The most effective direction of vectorization is across the vectors being transformed, and there are not many of these. The costs associated with the remaps, i.e., the interprocess communication required when changing between the latitude-longitude domain decomposition and the latitude-vertical decomposition, appear to be constant or slowly increasing for large process counts, though this is difficult to quantify due to the presence of the load imbalances. For 448 processors the remaps and the polar filters account for more than half of the time spent in the dynamics, and this percentage increases for larger processor counts. Note that the primary performance advantage of the X1E over the X1 in the dynamics is that this load imbalance issue affects performance on the X1 even more than on the X1E.

In contrast, the time spent in the polar filters and in the remaps is still decreasing from 448 processors to 840 processors on the XT3 and on the p575 cluster. Apparently the FFTs are fast enough compared to the computational rate for the rest of the dynamics that the load imbalance does not represent as much of a scalability bottleneck on these systems. However, more work is needed to quantify the exact nature of the scaling problem on the X1E and to explain the relative lack of a problem on the XT3 and p575 cluster.

Comparing the dynamics performance on the p575 cluster with performance on the XT3 when using 32 processors, both communication and computation appear to be faster on the p575 cluster, with the possible exception of the polar filters, but it is difficult to separate the communication cost from that of the computation. For larger processor counts a direct comparison of the profile data is problematic as the p575 cluster uses OpenMP and different domain decompositions than the XT3 for the same processor count. For example, for 840 processors the XT3 uses a 120x7 grid while the p575 cluster uses a 30x7 grid with 4 OpenMP threads per process. For a 3 day simulation, over half of the performance difference for 840 processors occurs in the remaps, so the advantage may be primarily lower communication overhead due to the use of OpenMP. However, for a 1 day simulation, both the remaps and the dynamics as a whole are faster on the XT3 than on the p575 cluster. This was not true for 32 processors, but as the processor count grows dynamics performance for 1 simulation day on the XT3 becomes steadily better relative to that of the p575 cluster while performance for 3 simulation days does not. At the very least, dynamics performance on the XT3 is more variable than on the p575 cluster. One conjecture is that diagnostic writes to standard output or standard error from within the dynamics is the source of the variability, but this has not been verified as of yet.

Figure 6.7 is a graph of seconds per simulation day for just the physics for the same five experimental platforms. Scalability is better for the physics than the dynamics for all platforms, though it is not perfect for large processor counts. With one exception, physics performance on the X1E is between 10% and 15% faster than on the X1. Compared to the p575 cluster, the X1E is between 2.6 and 3.2 times faster when using 256 processors or less, and approximately 2.4 times faster when using more than 256 processors. Performance on the XT3 is nearly identical to that on the p575 cluster up to 256 processors, remains within 6% up to 384 processors, but is slower by 15% for 448 processors and by 41% for 840 processors. The degradation in performance on the XT3 relative to that on the p575 cluster is due to the increasing percentage of time spent in a global sum and an associated write statement. This time grows with processor count, and takes approximately 5 times longer on the XT3 than the p575 cluster for all processor counts. For

840 processors this sum and write are 33% of the physics runtime on the XT3, but only 4% on the p575 cluster. Performance is nearly identical on the XT3 and p575 for most of the computational phases.

Figure 6.8 is a graph of seconds per simulation day for both dynamics and physics for the Cray X1E, Cray XT3, and IBM p575 cluster. For 0.5x0.625 L26, the dynamics is two to four times more expensive than the physics, just the reverse of the case for T85 L26. In the EUL dycore the physics is executed at the same frequency as the dynamics. In FV, the physics and dynamics use independent timesteps, and the physics is executed at a lower frequency than the dynamics.

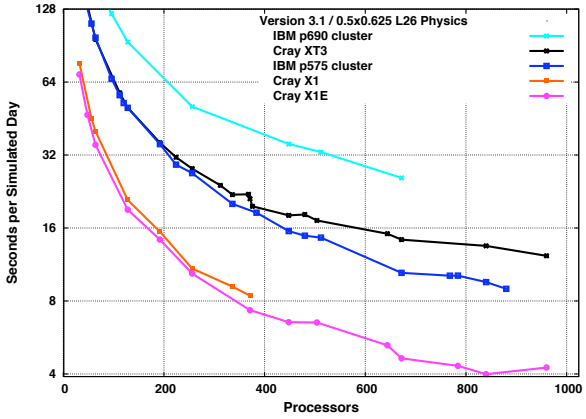

FIGURE 6.7: 0.5x0.625 L26 Physics

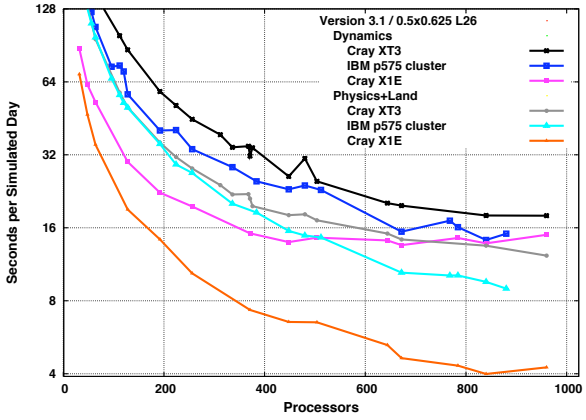

FIGURE 6.8: 0.5x0.625 L26 Diagnostics

7 Summary and Future Work

This paper described porting and optimizing CAM on the Cray X1E and Cray XT3 and performance for two CAM benchmarks. While there are a number of unresolved issues, we can state a few conclusions.

1. The Cray X1E is the fastest platform for CAM for these two benchmarks.
2. Performance on the Cray XT3 is similar to that on the IBM p575 cluster, but OpenMP parallelism gives the p575 cluster the advantage for large processor counts.
3. Performance on the Cray X1E is very sensitive to load imbalances, and monitoring performance is important as the code evolves.
4. Performance on the Cray XT3 is very sensitive to I/O.

There are a number of issues that will arise as CAM evolves, and it is important that we continue to improve the scalability of the code to take advantage of the next generation of petascale computing platforms. However, even for the current version of the code and for the current platforms there are a number of tasks.

1. On the Cray X1E we need to examine the polar filters and the remaps more closely, to quantify the exact nature of the performance problem and to investigate whether the calls to the FFTs and the remaps can be blocked to increase vectorization and decrease communication overhead.
2. On the XT3 we need to determine the source of the performance variability. If due to I/O, we need to determine if and when a write is necessary, and whether it can be buffered locally.

One of the modifications to CAM expected within the next year is the availability of a new FV dycore that uses a "cubed sphere" grid [18] instead of a longitude-latitude grid. Using such a grid will eliminate the need for the polar filters and the need to decompose the vertical dimension when using a two-dimensional domain decomposition. This will improve both performance and scalability on all of the systems, but is likely to have a larger impact on the Cray X1E and XT3 performance. Another planned change is adding the capability to run with a different number of (active) MPI processes in the physics and in the dynamics, providing some of the scalability of OpenMP without requiring OpenMP.

8 Acknowledgements

This research used resources (Cray X1, Cray X1E, Cray XT3, IBM p690 cluster, and SGI Altix) 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. It also used resources (IBM p575 cluster) of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. We thank A. Mirin at Lawrence Livermore National Laboratory for supplying CAM performance data on the Itanium2 cluster, D. Parks of NEC Solutions America for providing CAM performance data on the Earth Simulator, and M. Wehner of Lawrence Berkeley National Laboratory for providing CAM performance data on the IBM SP. We also gratefully acknowledge our collaborators, too numerous to mention here, in the performance engineering of CAM.

9 About the Author

Patrick H. Worley is a senior R&D staff member in the Computer Science and Mathematics Division of Oak Ridge National Laboratory. His research interests include parallel algorithm design and implementation (especially as applied to atmospheric and ocean simulation models) and the performance evaluation of parallel applications and computer systems. Worley has a PhD in computer science from Stanford University. He is a member of the Association for Computing Machinery and the Society for Industrial and Applied Mathematics.
E-mail: worleyph@ornl.gov.

References

[1] S. R. Alam, R. F. Barrett, M. R. Fahey, O. E. Messer, R. T. Mills, P. C. Roth, J. S. Vetter, and P. H. Worley, *Evaluation of the Cray XT3 at ORNL: a Status Report*, in Proceedings of the 48th Cray User Group Conference, May 8-11, 2006, R. Winget and K. Winget, ed., Eagan, MN, 2006, Cray User Group, Inc.

[2] M. B. Blackmon, B. Boville, F. Bryan, R. Dickinson, P. Gent, J. Kiehl, R. Moritz, D. Randall, J. Shukla, S. Solomon, G. Bonan, S. Doney, I. Fung, J. Hack, E. Hunke, and J. Hurrel, *The Community Climate System Model*, BAMS, 82 (2001), pp. 2357–2376.

[3] 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, to appear (2006).

[4] W. D. Collins, P. J. Rasch, and et. al., *Description of the NCAR Community Atmosphere Model (CAM 3.0)*, NCAR Tech Note NCAR/TN-464+STR, National Center for Atmospheric Research, Boulder, CO 80307, 2004.

[5] Community Climate System Model. http://www.ccsm.ucar.edu/.

[6] L. Dagum and R. Menon, *OpenMP: an industry-standard API for shared-memory programming*, IEEE Computational Science & Engineering, 5 (1998), pp. 46–55.

[7] J. B. Drake, P. H. Worley, I. Carpenter, and M. Cordery, *Experience with the Full CCSM*, in Proceedings of the 46th Cray User Group Conference, May 17-21, 2004, R. Winget and K. Winget, ed., Eagan, MN, 2004, Cray User Group, Inc.

[8] W. Gropp, M. Snir, B. Nitzberg, and E. Lusk, *MPI: The Complete Reference*, MIT Press, Boston, 1998. Second edition.

[9] J. T. Kiehl, J. J. Hack, G. Bonan, B. A. Boville, D. L. Williamson, and P. J. Rasch, *The National Center for Atmospheric Research Community Climate Model: CCM3*, J. Climate, 11 (1998), pp. 1131–1149.

[10] S.-J. Lin, *A 'vertically Lagrangian' finite-volume dynamical core for global models*, Mon. Wea. Rev., 132 (2004), pp. 2293–2307.

[11] Lustre. http://www.lustre.org/.

[12] Mathematical Acceleration Subsystem. www-306.ibm.com/software/awdtools/mass/.

[13] A. Mirin and W. B. Sawyer, *A scalable implemenation of a finite-volume dynamical core in the Community Atmosphere Model*, International Journal of High Performance Computing Applications, 19 (2005), pp. 203–212.

[14] National Center for Computational Sciences. http://www.nccs.gov.

[15] K. Oleson, Y. Dai, G. Bonan, M. Bosilovich, R. Dickinson, P. Dirmeyer, F. Hoffman, P. Houser, S. Levis, G.-Y. Niu, P. Thornton, M. Vertenstein, Z.-L. Yang, and X. Zeng, *Technical description of the Community Land Model (CLM)*, NCAR Tech Note NCAR/TN-461+STR, National Center for Atmospheric Research, Boulder, CO 80307, 2004.

[16] L. Oliker, J. Carter, M. Wehner, A. Canning, S. Ethier, A. Mirin, G. Bala, D. Parks, P. Worley, S. Kitawaki, and Y. Tsuda, *Leading Computational Methods on Scalar and Vector HEC Platforms*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC05), Nov. 12-18, 2005, IEEE Computer Society Press, Los Alamitos, CA, 2005.

[17] H. Pritchard, D. Gilmore, D. Knaak, M. Pagel, and M. ten Bruggencate, *Message Passing Toolkit (MPT) Software on XT3*, in Proceedings of the 48th Cray User Group Conference, May 8-11, 2006, R. Winget and K. Winget, ed., Eagan, MN, 2006, Cray User Group, Inc.

[18] M. Rancic, R. J. Purser, and F. Mesinger, *A global shallow water model using an expanded spherical cube: gnomonic versus conformal coordinates*, Quart. J. Roy. Meteor. Soc., 122 (1996), pp. 959–982.

[19] S. L. Scott, *Synchronization and Communication in the T3E Multiprocessor*, in Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems, October 1-5, 1996, New York, NY, 1996, Association for Computing Machinery, pp. 26–36.

[20] D. L. Williamson and J. G. Olson, *Climate simulations with a semi-lagrangian version of the NCAR Community Climate Model*, Mon. Wea. Rev., 122 (1994), pp. 1594–1610.

[21] P. H. Worley, *Benchmarking using the Community Atmosphere Model*, in Proceedings of the 2006 SPEC Benchmark Workshop, Jan. 23, 2006, Warrenton, VA, 2006, The Standard Performance Evaluation Corp.

[22] P. H. Worley and J. B. Drake, *Performance portability in the physical parameterizations of the Community Atmosphere Model*, International Journal of High Performance Computing Applications, 19 (2005), pp. 187–202.