LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-CONF-221711

Performance Engineering in the Community Atmosphere Model

P. Worley, A. Mirin, J. Drake, W. Sawyer

May 30, 2006

Scientific Discovery through Advanced Computing
Denver, CO, United States
June 25, 2006 through June 29, 2006

Updated October 14, 2003

Performance Engineering in the Community Atmosphere Model

P Worley[1], A Mirin[2], J Drake[1] and W Sawyer[3]

[1] Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, TN 37831-6016
[2] Lawrence Livermore National Laboratory, Livermore, CA 94551
[3] Swiss Federal Institue of Technology, 8092 Zurich, Switzerland

E-mail: worleyph@ornl.gov

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. Performance engineering has been an important aspect of CAM development throughout its existence. This paper briefly summarizes these efforts and their impacts over the past five years.

1. Introduction

The Community Climate System Model (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 [1]. CCSM development is the focus of the Department of Energy (DOE) Scientific Discovery through Advanced Computing (SciDAC) project *Collaborative Design and Development of the Community Climate System Model for Terascale Computers*. Performance and performance portability have been major foci of this project since its inception in July 2001.

The CCSM is constantly evolving to incorporate new science. The target computer platforms change periodically as well. The project performance engineering goals are addressed by an iterative process that includes porting the CCSM to new platforms, collecting and analyzing performance data, optimizing performance, and modifying code to improve performance portability. (Performance portability refers to the capability of a code to be optimized on a new platform or for a new problem instance quickly, and is an important enabler of this iterative process.) These activites are performed in close collaboration with the SciDAC Integrated Software Infrastructure Center on performance engineering *Performance Evaluation Research Center* (PERC). PERC provides both personnel and tools, and has been an important enabler of the success of the performance engineering activities, particularly in the areas of performance data collection and analysis.

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. This paper briefly summarizes the impact of the above mentioned SciDAC projects on both performance and performance portability of CAM over the past five years. For information on

performance engineering in other CCSM component models, see the special issue (volume 19, number 3) on climate modeling of the International Journal on High Performance Computer Applications.

2. Community Atmosphere Model

CAM is a global atmosphere model developed at the National Science Foundation's National Center for Atmospheric Research (NCAR) with contributions from researchers funded by the DOE and by the National Aeronautics and Space Administration (NASA) [2]. CAM is a mixed-mode parallel application code, using both the Message Passing Interface (MPI) [6] and OpenMP protocols [5]. 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 [3]. The approximations in the physics are referred to as the physical parameterizations. 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) [7], a spectral semi-Lagrangian (SLD) [12], and a finite volume semi-Lagrangian (FV) [8]. 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.

The development of CAM is a large community-wide effort with CAM software engineering led by the CCSM Software Engineering Group at NCAR. The authors were instrumental in much of the performance engineering of CAM [9, 14], including all three dycores and the physics. However, some of the performance portability features in the FV dycore were developed at NASA [10] and David Parks of NEC Solutions America, in partnership with the Japanese Earth Simulator Center, was responsible for the initial vectorization of many of the routines in CAM.

3. Performance Engineering

The general performance engineering goals are to (1) maximize single processor performance, e.g., optimize memory access patterns and maximize vectorization or other fine-grain parallelism, and (2) minimize parallel overhead, e.g., minimize communication costs, load imbalance, and redundant computation. These goals need to be achieved for a variety of target computer systems, a range of problem specifications, and a range of processor counts, all while preserving maintainability and extensibility. There is no optimal solution for all desired configurations of platform, problem, and processor count, and we rely on performance portability techniques. We have been successful in delaying decisions that affect performance until compile-time or run-time by hiding performance options in utility layers or in initialization routines. The code seen by developers has not been significantly impacted. The current set of CAM tuning options are discussed in [13].

Most of our optimization efforts can be categorized as either (1) eliminating unnecessary work, (2) cache blocking and/or vectorization, (3) exposing additional parallelism, (4) load balancing, (5) interprocessor communication optimizations, or (6) evaluating different compiler optimization options and exploiting optimized libraries. The largest impact on the code, in terms of number of lines modified, has come from exposing additional parallelism, e.g., by introducing a two dimensional domain decomposition where before there was only a one dimensional decomposition, and from enabling vectorization, e.g., by reordering loops and changing data structures to increase loop lengths.

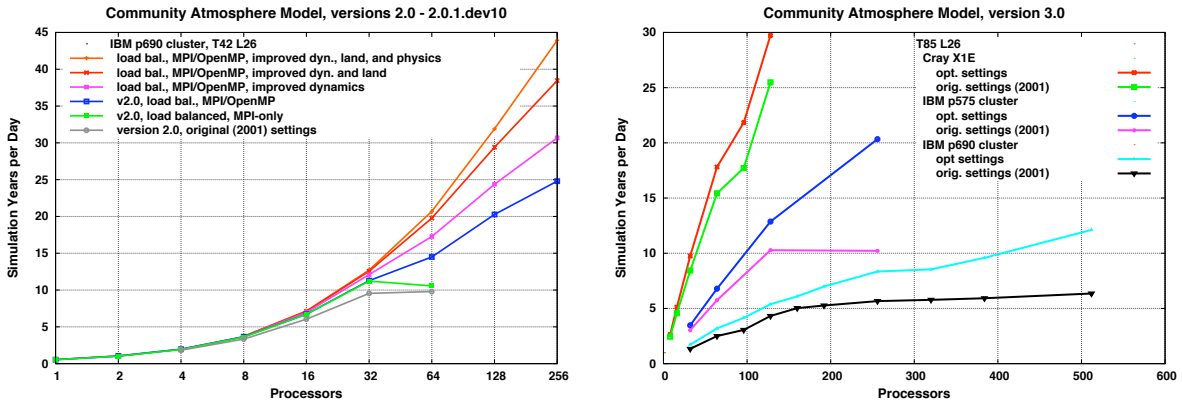

Figure 1. CAM EUL Performance History: June 2001 to May 2004.

4. Performance History

Documenting performance improvements in an evolving code is complicated. Changes that add, for example, new processes to the physics can increase the amount of work required in a simulation. Such complexity-changing modifications invalidate performance optimization comparisons between versions of the code before and after the modification. Periodically, the problem specification or dynamical core of most interest also changes, which requires the redefinition of baseline performance using the current production version of the code. (It is often not possible to run the new problem specification on older versions of the code.) A version control system is essential for this type of study as it documents the code evolution. First CVS, then Subversion [4], have been used with CAM. One side effect of introducing performance optimizations as compile-time and run-time options is that new versions of the code can be run in the "old way", so that the performance impact of the introduction of a performance tuning option can still be quantified.

The left graph in Figure 1 describes the performance improvement between June 2001 and November of 2002 on an IBM p690 cluster (32-way symmetric multiprocessor (SMP) nodes with 1.3 GHz Power4 processors and IBM SP Switch2 interconnect between nodes). The benchmark problem T42 L26 uses a 64x128x26 (latitude by longitude by vertical) computational grid and the EUL dycore, which were the production settings at the time of the experiments. The initial curve is the performance when setting the optimization options to emulate the way that CAM version 1.0 was run in June 2001. Note that the performance improvements include increasing the number of processors that could be used effectively from 64 to more than 256.

The right graph in Figure 1 describes the performance improvement due to the performance optimization options introduced between June 2001 and May 2004 on the IBM p690 cluster, now using the IBM HFS interconnect, on an IBM p575 cluster (8-way SMP nodes with 1.9 GHz Power5 processors and IBM HFS interconnect between nodes), and on a Cray X1E (cluster of 4-way SMP nodes with a 2D torus interconnect where each node contains four 1.13 GHz 8-pipe vector processors). Results are for the new production problem size of T85 L26, which uses the EUL dycore and a computational grid of size 128x256x26.

So far we have described the impact of performance optimizations in the physics and in the spectral dycores. Over the same period of time the NASA FV dycore was integrated with CAM, a two-dimensional (2D) domain decomposition option was implemented, and a number of FV communication optimization options were added. As FV was not available in CAM previously, no baseline existed against which to evaluate these optimizations, and many of the optimizations were introduced simultaneously. The left graph in Figure 2 compares CAM performance when using MPI-2 one-sided and MPI-1 two-sided communication requests with varying levels of

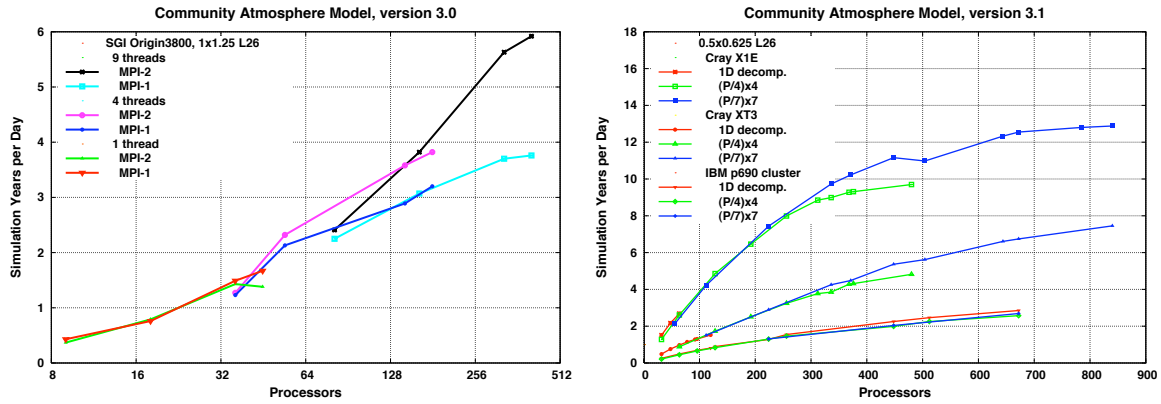

Figure 2. CAM FV Performance Optimizations.

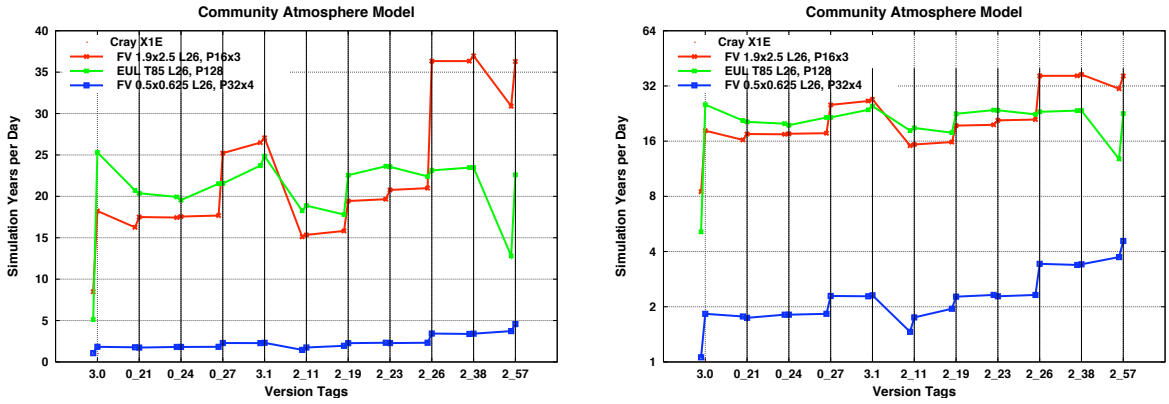

Figure 3. CAM Vector Performance History: March 2005 to March 2006.

thread parallelism. The experimental platform is an SGI Origin3800 nonuniform memory access global shared memory system with 600 MHz MIPS R14000 processors. The benchmark problem (1x1.25 L26) uses a 181x288x26 computational grid.

The right graph in Figure 2 illustrates the performance impact of the 2D domain decomposition on performance on three platforms, the Cray X1E, the Cray XT3, and the IBM p690 cluster. The Cray XT3 is a cluster of single processor nodes with 2.4 GHz Opteron processors and a 3D torus interconnect. The benchmark problem (0.5x0.625 L26) uses a 361x576x26 computational grid. Performance is graphed for the original one-dimensional (1D) decomposition, a 2D decomposition where four processors are applied to the new dimension, and a 2D decomposition in which seven processors are applied to the new dimension. The 1D decomposition is limited to 120 MPI processes for this benchmark. The 2D domain decomposition increases MPI scalability significantly, but with diminishing performance returns for high processor counts. For the IBM p690 cluster, OpenMP parallelism is more efficient at increasing scalability than is the 2D decomposition up to the indicated number of processors.

The most recent performance engineering efforts involved (re)vectorizing CAM without degrading performance on the nonvector systems. The target vector systems are the Cray X1E and the Earth Simulator. As these two systems have somewhat different performance sensitivities, maintaining performance portability between them has been an issue. Figure 3 illustrates the performance history for three benchmark problems and processor counts on the X1E from May 2005 to May 2006. The versions named on the X-axis were developed on the

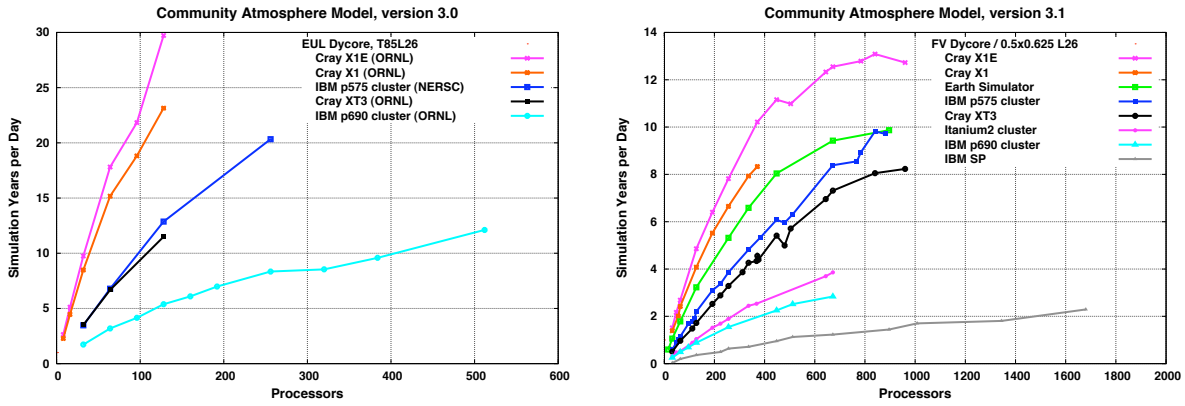

Figure 4. CAM Performance, May 2006

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. The two graphs contain the same data, but the one on the right uses a logarithmic Y-axis. From this it should be clear that maintaining vectorization is a significant performance advantage, and requires ongoing monitoring as new code is introduced. The new benchmark problem, 1.9x2.5 L26, has a 96x144x26 computational grid and is the initial production resolution for the FV dycore within CCSM.

5. Future Challenges

As shown in section 3, performance engineering efforts over the past five years have improved CAM performance significantly. The graphs in Figure 4 describe current CAM performance, where recent performance optimizations have been backported into versions 3.0 and 3.1. While performance on the current production platforms is very good, scalability is still limited. CAM is evolving quickly at the current time. Computational complexity, load imbalance in the physics, and communication overhead in the dynamics are all expected to increase significantly. To maintain the required simulation throughput rates will require further improvement in processor scalability and algorithm flexibility, and performance portability techniques will continue to be vital to achieving CAM performance goals.

6. Acknowledgements

We thank 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.

This research used resources (Cray X1E, Cray XT3, IBM p690 cluster) 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 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.

The work of Drake and Worley was supported by 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. The work of Mirin was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48. The work of Sawyer was performed under the auspices of the NASA Goddard Space Flight Center, NASA Task 00-9103-01a. 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.

References

[1] 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.

[2] 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(11) (2006).

[3] 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.

[4] B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato, *Version Control with Subversion*, O'Reilly Media, Inc., Sebastopol, CA, June 2004.

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

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

[7] 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.

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

[9] 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.

[10] W. Putman, S. J. Lin, and B. Shen, *Cross-platform performance of a portable communication module and the NASA finite volume general circulation model*, International Journal of High Performance Computing Applications, 19 (2005), pp. 213–224.

[11] M. Rancic, R. J. Purser, and F. Mesinger, *A global shallow-water model using an expanded spherical cube: gnomic versus conformal coordinates*, Q. J. R. Met. Soc., 122: 959-982 (1996).

[12] 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.

[13] 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.

[14] 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.