Porting and Performance of the Community Climate System Model (CCSM3) on the Cray X1

George R Carr Jr, *NCAR;* **Matthew J Cordery**, *Cray;*
John B Drake, Michael W Ham, Forrest M Hoffman,
Patrick H Worley, *ORNL*

ABSTRACT: *The Community Climate System Model (CCSM3) is the primary model for global climate research in the United States and is supported on a variety of computer systems. We present some of our porting experiences and describe the current performance of the CCSM3 on the Cray X1. We include the status of work in progress on other systems in the Cray product line.*

KEYWORDS: CCSM, climate, Cray X1, porting, performance.

1. Introduction: The Community Climate System Model

The Community Climate System Model (CCSM3) is a computer model for simulating the Earth's climate. The CCSM is supported by the National Science Foundation and the Department Of Energy and is freely available to the climate community.

The CCSM is built from four dynamical component models: atmosphere, ocean, land surface and sea ice. These communicate with each other via a flux coupler component in a "hub and spoke" configuration (see Figure 1). The CCSM and its components are fully documented in papers available from web pages at the National Center for Atmospheric Research (NCAR) (http://www.ccsm.ucar.edu/models/ccsm3.0) and other papers [Special Issue on Climate Modeling, Int'l J. HPC Apps.,Vol 19, #3, August, 2005] [CCSM Special Issue, J. Climate, 11(6)] [Collins, 2005].

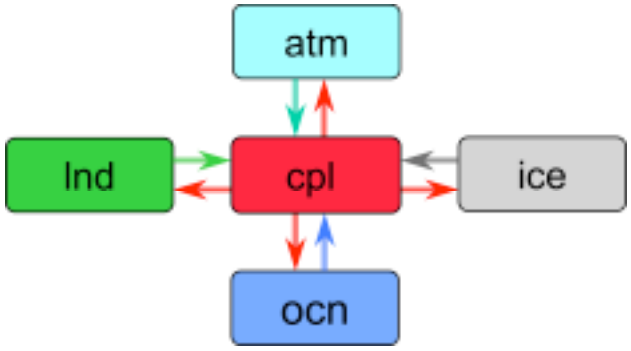

Figure 1 Hub and Spoke Design

1.1 The Components

The atmospheric component of CCSM3 is the Community Atmosphere Model (CAM3), a descendant of the NCAR atmospheric climate models [Washington, 1982] [Williamson, 1983]. Standard resolutions are T85 for 1.4 degree horizontal grid spacing (128x256x26 grid size), T42 for 2.8 degree (64x128x26), and T31 for 3.75 degree (48x96x26). The 26 level vertical grid uses a hybrid pressure coordinate system [Collins, 2004].

The ocean component is based on the Parallel Ocean Program (POP), version 1.4.3 [Smith and Gent, 2002], an ocean circulation model developed at Los Alamos National Laboratory. Typical resolutions are one degree in the horizontal (320x384x40) and three degree (100x116x25).

The land component is the Community Land Model (CLM3). The land component operates on the same horizontal grid as the atmospheric component [Levis, 2004] [Oleson, 2004].

The sea ice model CSIM5 is based on the Los Alamos CICE. The ice model uses the same horizontal grid as the ocean component [Briegleb, 2004].

The flux coupler (CPL6) performs a number of the data and grid conversions required to pass data from component to component [Craig, 2005].

1.2 CCSM Configurations

The CCSM3 can be run with a variety of options for each. For example, the CCSM3 currently supports options for the ocean model ranging from a "Data" model that is often used for testing the CCSM3 to a somewhat simplified slab ocean model to the use of a complete ocean model (POP).

In this paper, all reference to CCSM configurations refer to "fully coupled" options that utilize CAM3, POP1.4.3, CLM3, CSIM5, and CPL6.

The CCSM is run on a large number of computer architectures, ranging from a workstation class multiprocessor to clusters of numerous varieties to vector supercomputers. The need to be able to run on all of these machines guarantees that performance is compromised for most if not all machines. Some aspects of the implementation of the model allow for accommodation to key aspects of the (very different) architectures.

The current version of CCSM is derived from contributions from researchers around the world over a period of more than 25 years. The code base is written in multiple versions of Fortran and a small amount of C. The CCSM requires the use of MPI and allows the additional use of OpenMP on some architectures. This, together with being a Multiple Program Multiple Data (MPMD) application, is often a significant challenge for vendors and application porters alike.

2. Introduction: Cray X1

The Cray X1 architecture combines vector and scalar processor units with cache and a globally addressable memory. The Single Streaming Processor (SSP) is composed of a single scalar unit and two vector pipes. The Multi-Streaming Processor (MSP) is composed of 4 SSPs ganged together with a 2MB cache. A node board contains 4 MSPs. Memory is physically distributed but globally addressable. At this time, the CCSM is using MSPs as processor units. The CCSM does not make use of OpenMP on the X1 at this time but does use some of the Cray Streaming Directives (CSDs) to gain similar effects. In this way, the CCSM relies on the compiler to spread the computation across the 8 vector pipes of each MSP.

3. CCSM port and validation process

3.1 Porting Introduction

The process of porting a code of the size and complexity of the CCSM is quite involved. Just getting it to build the first time on a new machine can be several weeks of work. Generation of correct climatic results requires much more work and time. When porting to a new machine, the safe approach is to begin with limited or no compiler optimization. The CCSM regularly finds a number of bugs in a new compiler that may require a number of workarounds and compiler fixes. With the Cray X1, for example, Programming Environment 5.2 was skipped entirely due to various issues. Because the CCSM has the extra complication of being an MPMD application, one usually starts with the standalone version of CAM.

3.2 CAM PERGRO

Once standalone CAM appears to be running properly, a perturbation growth test (PERGRO) can quickly tell if the numeric results are within a reasonable bound. A complete description of the CAM port validation process, including the PERGRO test, can be found at http://www.ccsm.ucar.edu/models/atm-cam/port/. The two simulation day computational requirement for the PERGRO test is very small. The PERGRO test simply shows the effect of roundoff error on the computation. A test of a new compiler or compiler options will hopefully differ from a base climate run by no more than the roundoff differences from a base validated climate run. If the difference is dramatically different than this simple test then significant problems exist and more extensive testing is delayed until the problems are resolved.

Figure 2 shows an example of several runs. Results from 4 runs are plotted in the figure. The curves for runs 2 and 3 coincide in the middle of the figure. The black curve at the bottom shows the impact of roundoff errors on the solution when calculated on the IBM Power4 machine at NCAR. The green line that hugs the left and upper axis shows a compiler and set of options that are clearly producing incorrect results, being significantly different than the solution calculated on the IBM. The purplish pair of plots in the middle show results that differ from the IBM results by more than roundoff but which cannot be rejected immediately. Further testing is required.

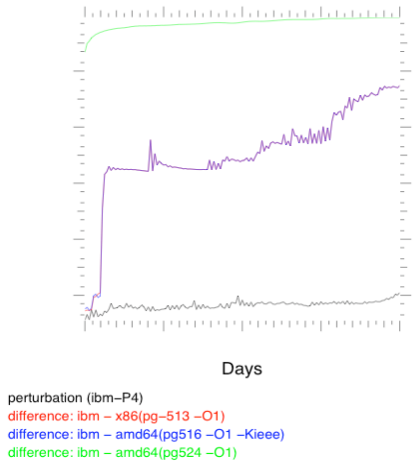

Figure 2: CAM Perturbation Growth Test

3.3 CAM/CCSM Atmospheric Diagnostics

The next step is typically to compare the monthly history files generated by CAM against a 100 year baseline. This test computes the monthly averages for the test configuration and the baseline and generates a large number of plots and graphs for analysis. A duration of the test configuration that is less than 100 years can be compared against the 100 years of the baseline and may be enough to show numeric divergence. A complete 100 years of the test configuration is required to accept the test configuration as valid. These atmospheric diagnostics can be made comparing data from the standalone CAM to a standalone CAM baseline or by comparing data from CCSM CAM to a CCSM CAM baseline. Often, this test is performed as a sanity check along the way while running 100 years of a controlled CCSM run for the CCSM Statistical Test discussed in the next section. Divergence of the results permits stopping the test configuration prior to completing 100 years of simulation.

The ocean, land, and sea ice components also have specific tests that can be run to aid in isolation of problems.

3.4 CCSM Statistical Analysis

The final step in the validation of a CCSM port is a statistical analysis of the CAM history files generated from a run of 100 simulation years of CCSM. Generating 100 years of data can take considerable computer time. On the Cray X1, running 100 years of the small T31x3 resolution model on 40 MSPs continuously currently takes more than 2.5 days to complete. Running 100 years of a much larger T85x1 resolution on 132 MSPs continuously currently takes more than 9 days.

3.5 CCSM Regression Testing

Once an acceptable baseline exists for a machine, simple tests can be performed to test that a change produces "bit for bit" exactly the same results. Code changes generating results that are not bit for bit must go through the full validation process to be accepted into the code baseline.

4. Some Aspects of the CCSM

4.1 CCSM Performance

The production performance of the CCSM3 is most often expressed as production throughput in number of simulated years per wall clock day for a specified number of processors (or years per day). A century long simulation takes 25 days for a computer delivering 4 years per day. Scaling efficiency is expressed as simulated years per wall clock day per CPU (or years per day per cpu). Table 1 shows the performance on each computing platform of the standard IPCC (T85 atmosphere, 1 degree ocean) model configuration. The number of processors used for a production run is a choice based on load balance of the components, batch queue constraints, and an estimate of the time required to generate the results. The turn around time can be measured in weeks when a large simulation of a thousand years or more are computed [Drake, 2005].

Platform	IBM SP3	IBM p690	ES(NEC SX6)	Cray X1
Num CPUs	208	192	184	208
Years/day	1.57	3.43	16.2	13.6
Years/day/cpu	0.0075	0.0179	0.0880	0.0654

Table 1: Computational Performance of CCSM3.0 for an IPCC T85x1 Run

4.2 Processor Load Imbalance

A primary computational challenge in the CCSM is the load imbalance generated by the non-homogeneous structure of a multi-physics, multi-component model. A striking example of the structure of the load imbalance appears in the calculation of the short wave radiation

balance. This computation need be done only where the sun is shining, i.e. on half of the computational domain. This region changes for each time step. Load imbalances within a component are typically resolved using data decomposition schemes such as those discussed in the next section.

Load imbalances are also generated from the concurrent component execution model used by CCSM. CCSM launches five individual binaries that run concurrently on separate processor sets. Each of the four dynamic components communicates with each other via the coupler component at prescribed stages of processing. Choosing a "correct" number of processors for each component is at best a compromise. The goal for a specified total number of processors is to provide a number of processors for each component such that processing delays are minimized, idle processor time is minimized, and the maximum simulation years per day is achieved. This is complicated as each component has different scaling attributes and different data decomposition restrictions. Some component processing is dependent on other component processing. A poorly chosen assignment of processors may result in one component waiting excessively on the results from another.

Typically, for a fully active T85x1 configuration, two of every 3 processors of the total processor count are assigned to the atmospheric component. The number of processors assigned to the ocean component is chosen to best match the processing time of the atmosphere. The balance of the processors are assigned to the sea ice, land, and coupler, with the goal being to keep the more numerous atmospheric processors busy.

Figure 3 shows raw performance for a number of machines using the T31x3 resolution. Clusters of results show that small changes in the assignment of processors to components can greatly affect the performance of the configuration.

Figure 4 shows the efficiency of each load balance experiment. A horizontal line would indicate a machine with perfect scaling.

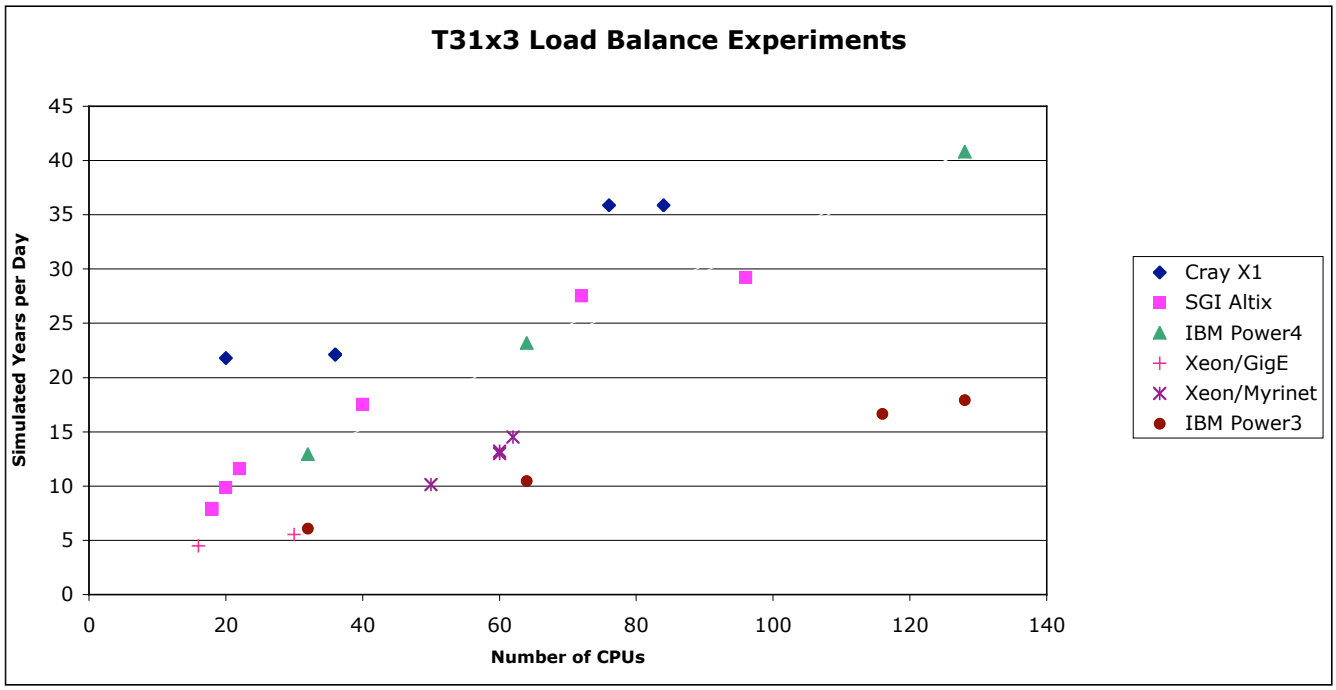

Figure 3: Load Balance Experiments (Performance)

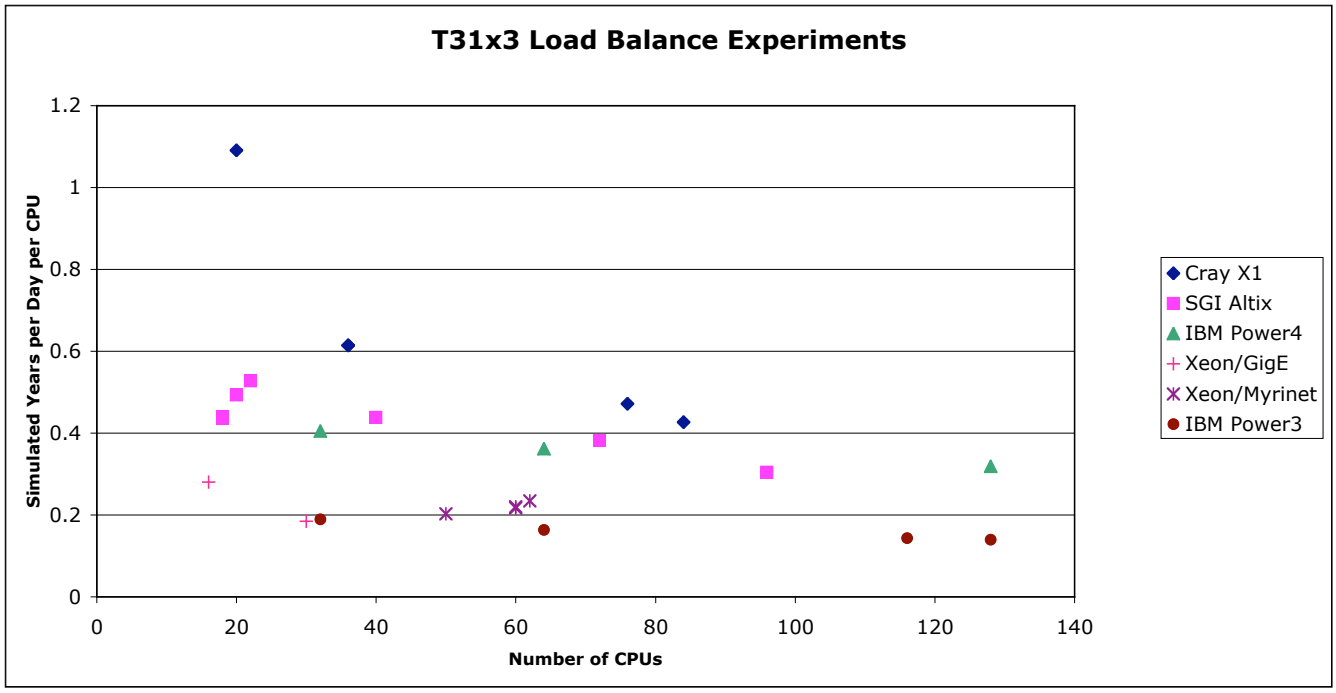

Figure 4: Load Balance Experiments (Efficiency)

Note: the data in Figures 3 and 4 do not include the latest (not yet validated) performance for the Cray X1. See below for projections.

5. Port Status of the CCSM on the Cray X1

5.1 Fall, 2004

Last fall the first successful T31x3 validation on the Cray X1 was completed with programming environment 5.1.0.5 (PE5105). With a normal conservative first try approach, the performance of the T31x3 runs with this baseline was only adequate and scaling was acceptable. The performance and scaling of the T85x1 runs were not quite as good as with the T31x3 runs. However, science could now be performed on the X1. The primary goal was to achieve correct scientific results. Performance was a secondary priority.

5.2 Winter, 2004/2005

A number of changes were proposed and tested including using the newer programming environment 5.3.0.2 (PE5302), different compiler options, different Cray Streaming Directives, use of a different option for CAM load balancing of the physics computations, and use of a different CAM value controlling vector length, allowing better vector performance. The combination of these were thought to be safe. Some were known to be bit for bit with PE5105. Others were expected to be within roundoff. Unfortunately, the tests all passed except for the final 100 year statistical test.

5.3 Current Work

The use of the compiler option –Ofp1 was not one of the compiler flags used with our initial validation. With the newer PE5302 the use of –Ofp1 proved necessary to generate correct results.

A minor difference between the two environments is the use of NOMODINLINE (disabling the automatic inlining of routines found in modules). With PE5105, the default was to turn on the NOMODINLINE option. With PE5302, the default was to turn off the NOMODINLINE option. This also perturbed the results. For now, the build procedures are set to specify NOMODINLINE.

Attempts were made to use the compiler options vector3, scalar3, and stream3 to speed up CCSM. (The defaults are vector2, scalar2, and stream2.) Use of these 3 options with all of CAM proved to perturb the results too much. However, use of the options with all routines of the CLM appears to be acceptable at this stage of testing. Current plans are to evaluate using these compiler options with the other components and with 4 specific CAM routines shown by profiling to be important.

CAM supports a number of options for physics decomposition. Where usable, these options can reduce load imbalance for this phase of the CAM processing. The speed of the Cray X1 interconnect is fast enough for this to be a major win. A significant amount of work was spent in the implementation of these load balance options

to assure that results would be bit for bit. Control of this is a runtime option [Worley, 2005].

Another feature of CAM and CLM is the ability to specify a work unit that controls the array length of some of the primary work units. Control of this permits setting sizes appropriate for cache machines or vector machines. A change in the value for CAM improved the performance on the Cray X1. A change for CLM is still being evaluated, as is an alternative use of Cray Streaming Directives (CSDs).

6. Projected Performance

The performance of the failed validation is shown below in figures 5 and 6. We anticipate that the performance of our next baseline will approach these numbers (but may fall a bit short). The primary sources for the improved performance, the CAM modifications, have been shown to produce "bit for bit" results in our current testing. The next validation attempt is just beginning.

Figure 5 shows the anticpated performance improvement for the T31x3 scenario. Figure 6 shows the anticipated improvement for the much larger resolution T85x1 scenario.

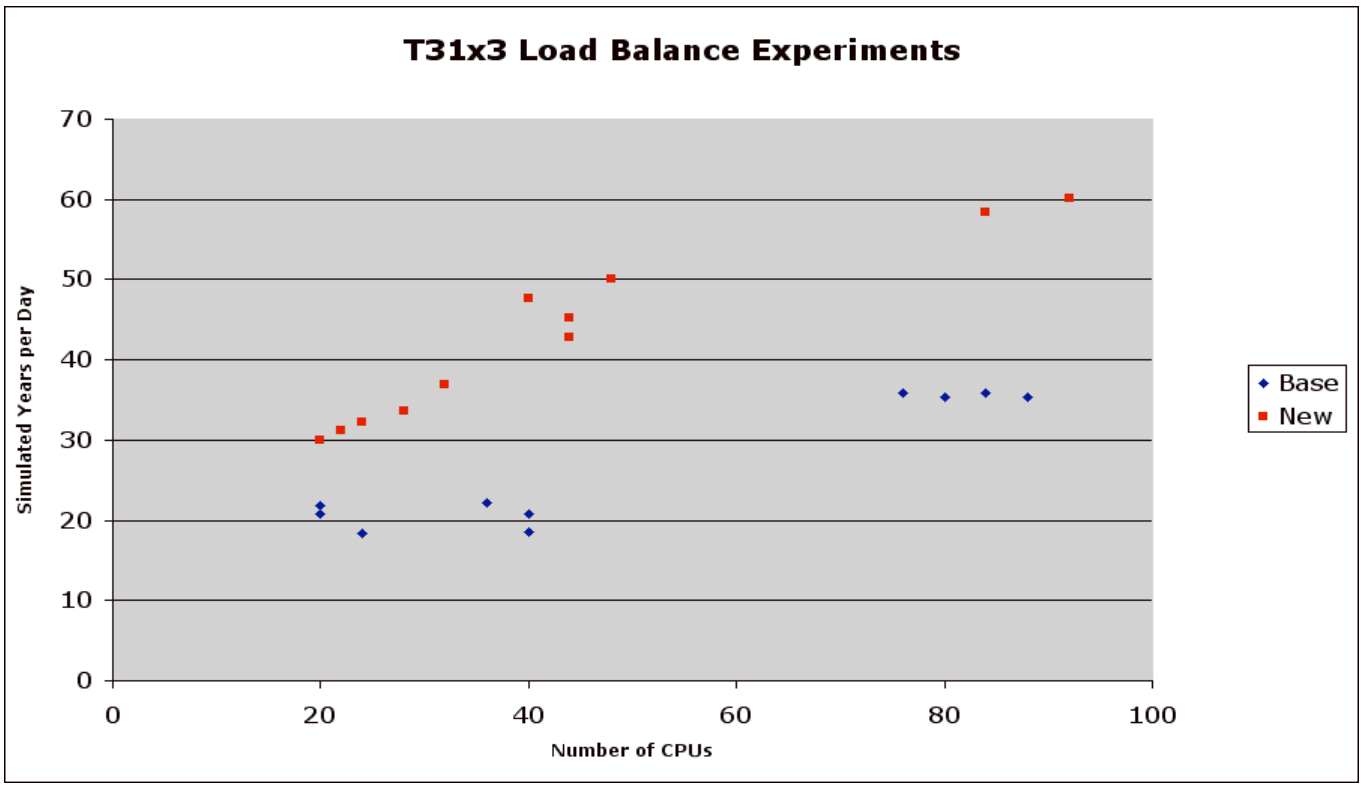

Figure 5: T31x1 Performance Estimates

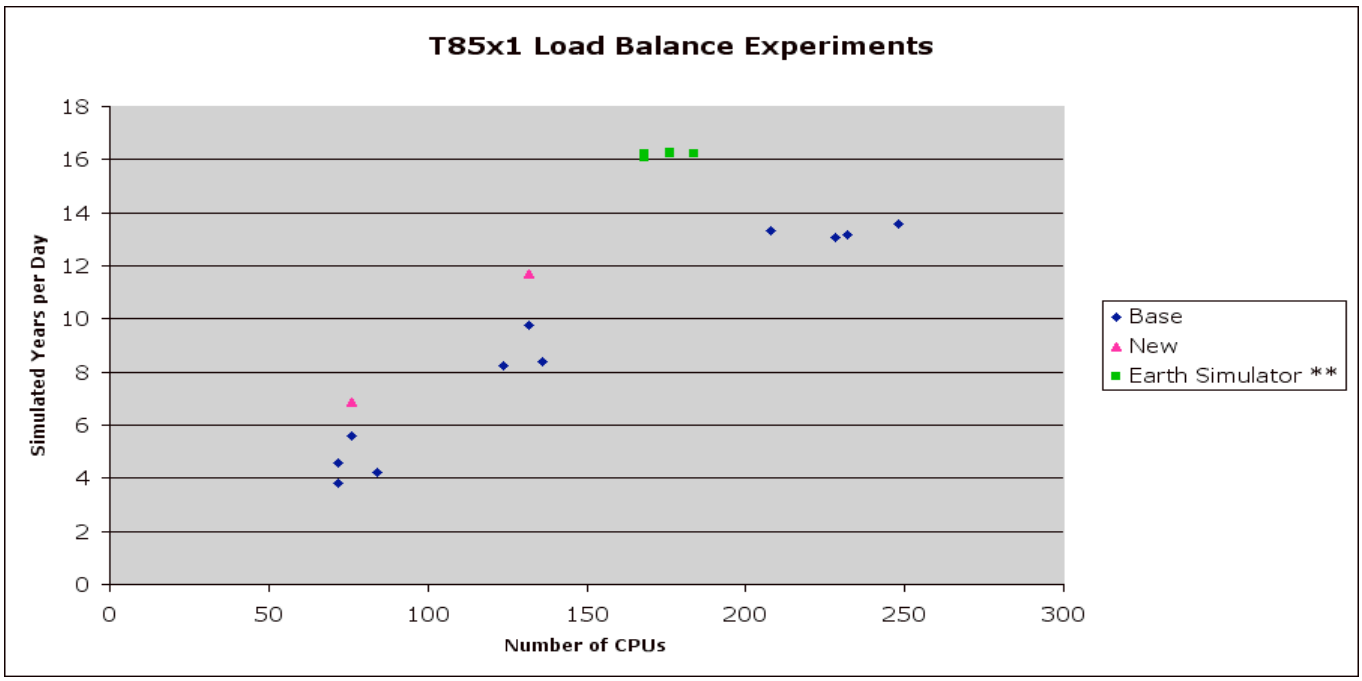

Figure 6: T85x1 Performance Estimates

The improvement measured for the T85x1 is not as large as that for the T31x3. One possible explanation is that the compiler was achieving better performance with the larger array sizes of the T85x1 than it could with the smaller T31x3. Further analysis with the Cray loop marks may assist in this analysis.

7. Remaining Work

A T31x3 validation attempt with the majority of the proposed changes is to begin immediately. Upon successful completion, a T85x1 IPCC validation will be performed. It is our goal to get the code into the hands of the CCSM community as soon as we can to support the ongoing science and model development activities.

The primary goal at this point is to improve the general performance and scalability of the model. Previous performance improvements have been made to the physical components. It now appears that general gains may be available through improvements in the coupler.

A new process for timing performance was introduced into CAM recently, and may be introduced into the rest of the CCSM in the near future. This will be used to generate a more complete performance characterization on several key machines, including the Cray X1. This will allow better focus on areas with the greatest potential gain. It will also facilitate automated performance regression testing. This will be key to targeting and fixing problem areas of the CCSM and maintaining good overall performance.

Work has begun on porting CCSM to the Cray XT3 and to the Cray XD1. At this point, issues have been encountered with regard to compilation and basic execution. Launching an MPMD application has been an issue on the Cray XD1.

Cray will continue to bring out new software environments that will need to be put through a full validation test. It is likely that most PE upgrades will not result in bit for bit results. Work has begun with programming environment 5.4.

Conclusion

Significant progress has been made improving the performance of the CCSM on the Cray X1. A number of methods of supporting portable performance through configuration options have proven useful on the Cray X1. The Cray X1 is becoming a significant resource for the CCSM community.

Acknowledgments

The research and development activities of Carr at NCAR were sponsored by the National Science Foundation. There are far too many at NCAR who contributed to George's work to be able to list them all here. Special thanks to Lawrence Buja, Bill Collins, Jim Hack, Tom Henderson, Mark Moore, Phil Rasch, and Mariana Vertenstein for their support, encouragement, and tolerance.

The research and development activities of Drake, Ham, Hoffman, and Worley were sponsored by the Climate Change Research Division of the Office of Biological and Environmental Research, 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.

About the Authors

George R Carr Jr is a Software Engineer at the National Center for Atmospheric Research (NCAR) in Boulder, CO. He has worked with many of the high performance computing platforms performing systems engineering and systems architecture analysis in a variety of applications areas including signal and image processing, numeric weather prediction, and climate modeling. He is currently responsible for the portability and performance of the Community Climate System Model (CCSM). He received a Bachelor of Arts degree in Mathematics at Drake University and a Master of Sciences degree in Computer Science at The Ohio State University. He is a member of the ACM, the IEEE Computer Society, and USENIX. George can be reached at NCAR, P.O. Box 3000, Boulder, CO 80307-3000, USA, E-mail: gcarr@ucar.edu.

Matthew Cordery is a Software Engineer for Cray Inc, where he specializes in optimizing weather and climate applications. He has a PhD in marine geology and geophysics from MIT and the Woods Hole Oceanographic Institution. He can be reached at Cray, Inc., 411 1st Ave S, Suite 600, Seattle, WA 98104 USA, E-mail: mcordery@cray.com.

John Drake is the Climate Dynamics Group Leader at the Oak Ridge National Laboratory. He received his Ph.D. in Mathematics from the University of Tennessee in 1991 and has worked on projects involving fluid dynamics simulation among the labs of the Oak Ridge complex since 1979. His primary research interest is in numerical algorithms for atmospheric flows that involve problems of scale and closure. Parallel algorithms and high-end computational techniques have been the focus of several recent projects to develop state of the art climate models for parallel computers. He can be reached at Oak Ridge National Laboratory, PO Box 2008, Oak Ridge, Tennessee, 37831-6016, USA, Email: drakejb@ornl.gov.

Michael W. Ham is a researcher at Oak Ridge National Laboratory in the Computer Science and Mathematics Division. He received a Bachelor of Science degree from The Pennsylvania State University and a Master of Science degree from the University of Illinois at Urbana-Champaign. He is interested in maximizing the performance and correctness of large scientific applications, particularly those related to weather and climate. He can be reached at Oak Ridge National Laboratory, PO Box 2008, Oak Ridge, TN 37831-6016, USA, E-mail: hammw@ornl.gov.

Forrest Hoffman is a researcher at Oak Ridge National Laboratory where he holds joint appointments in the Computer Science & Mathematics and the Environmental Sciences Divisions. He received Masters and Bachelors of Science degrees in physics from the University of Tennessee. Forrest performs computational science research in global climate, landscape ecology, and terrestrial biogeochemistry on Linux clusters as well as some of the world's largest supercomputers in the National Center for Computational Sciences at ORNL. He can be reached at Oak Ridge National Laboratory, P.O.Box 2008 MS6008, Oak Ridge, TN 37831-6008, USA, Email: forrest@climate.ornl.gov.

Patrick H. Worley is a Senior Research Scientist in the Computer Science and Mathematics Division at Oak Ridge National Laboratory. His research interests include parallel algorithm design and implementation, the performance evaluation of high performance computer systems, the performance evaluation and optimization of parallel scientific applications, and the numerical simulation of PDEs. Worley has a Ph.D. in computer science from Stanford University. He is a member of SIAM and the ACM. He can be reached at Oak Ridge National Laboratory, P.O.Box 2008 MS6016, Oak Ridge, TN 37831-6016, USA, Email: worleyph@ornl.gov.

References

Briegleb, B. P., C. M. Bitz, *E. C. Hunke, *W. H. Lipscomb, M. M. Holland, J. L. Schramm, and R. E. Moritz, 2004. Scientific description of the sea ice component in the Community Climate System Model, Version Three. NCAR Tech. Note NCAR/TN-463+STR, 70 pp

Collins, W.D, C. M. Bitz, M. L. Blackmon, G. B . Bonan, C. S. Bretherton, J. A. Carton, P. Chang, S. C. Doney, J. J. Hack, T. B. Henderson, J. T. Kiehl, W. G. Large, D. S. McKenna, B. D. Santer, and R. D. Smith, 2005. The Community Climate System Model: CCSM3 to be published in the CCSM Special Issue, J. Climate, **11(6)**, to appear

Collins, W. D., P. J. Rasch, B. A. Boville, J. J. Hack, J. R. McCaa, D. L. Williamson, J. T. Kiehl, B. Briegleb, C. Bitz, *S.-J. Lin, M. Zhang, and Y. Dai, 2004. Description of the NCAR Community Atmosphere Model (CAM 3.0). NCAR Tech. Note NCAR/TN-464+STR, p. 226.

Craig, A. P., R L Jacob, B Kauffman, T Bettge, J Larson, E Ong, C Ding, Y He. Cpl6: The New Extensible, High-Performance Parallel Couler for the Community Climate System Model. Submitted for publication

Special Issue on Climate Modeling, Int'l J. HPC Apps.,Vol 19, #3, August, 2005

Drake, J. B., P W Jones, G R Carr Jr. Overview of the Software Design of the CCSM. Submitted for publication Special Issue on Climate Modeling, Int'l J. HPC Apps.,Vol 19, #3, August, 2005

Levis, S., G. B. Bonan, M. Vertenstein, and K. W. Oleson, 2004. The Community Land Model's Dynamic Global Vegetation Model (CLM-DGVM): Technical description and user's guide. NCAR Tech. Note NCAR/TN-459+IA, p. 50.

Oleson, K. W., 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, 2004. Technical description of the Community Land Model (CLM). NCAR Tech. Note NCAR/TN-461+STR, 174 pp

Smith, R.D. and P. Gent 2002. Reference manual for the Parallel Ocean Program (POP). Los Alamos Unclassified Report LA-UR-02-2484.

Washington, W.M., 1982. Documentation for the Community Climate Model (CCM) Version 0. NCAR report, Boulder Colorado, NTIS No. PB82 194192

Williamson, D.L., 1983. Description of the NCAR Community Climate Model (CCM0B). NCAR Technical Note NCAR/TN-210+STR, Boulder Colorado, NTIS No. PB83 231068

Worley, P.H. and J.B. Drake, 2005. Performance Portability in the Physical Parameterizations of the Community Atmospheric Model. Submitted for publication Special Issue on Climate Modeling, Int'l J. HPC Apps.,Vol 19, #3, August, 2005