OVERVIEW OF THE SOFTWARE DESIGN OF THE COMMUNITY CLIMATE SYSTEM MODEL

John B. Drake[1]
Philip W. Jones[2]
George R. Carr, Jr.[3]

Abstract

The Community Climate System Model (CCSM) is a computer model for simulating the Earth's climate. The CCSM is built from four individual component models for the atmosphere, ocean, land surface, and sea ice. The notion of a physical/dynamical component of the climate system translates directly to the software component structure. Software design of the CCSM is focused on the goals of modularity, extensibility, and performance portability. These goals are met at both the component level and within the individual component models. Performance portability is the ability of a code to achieve good performance across a variety of computer architectures while maintaining a single source code. As a community model, the CCSM must run on a variety of machine architectures and must perform well on all these architectures for computationally intensive climate simulations.

Key words: Parallel clusters, multiphysics simulation

The International Journal of High Performance Computing Applications, Volume 19, No. 3, Summer 2005, pp. 177–186
DOI: 10.1177/1094342005056094

1 Introduction: Community Climate System Model

The Community Climate System Model (CCSM) is a computer model for simulating the Earth's climate. It is supported primarily by the United States National Science Foundation (NSF) and the United States Department of Energy (DOE) and is freely available to the climate community for use in climate research, climate prediction, and assessment of climate change scenarios. The CCSM, like other coupled climate models, is built from four individual component models for the atmosphere, ocean, land surface, and sea ice. The notion of a physical/dynamical component of the climate system translates directly to the software component structure. The physical basis for coupling the models has been worked out through the development of a flux coupler component, which takes into account the different natural time-scales for each component as well as the need to conserve physical quantities such as mass, energy, and momentum in the coupled system (Collins et al., 2005b). All codes are fully documented and detailed descriptions are available at http://www.ccsm.ucar.edu/models/ccsm3.0.

The atmospheric component of CCSM3 is the Community Atmosphere Model (CAM) and is a descendant of the National Center for Atmospheric Research (NCAR) atmospheric climate models (Washington, 1982; Williamson, 1983). Standard resolutions are T85 for 1.4 degrees ($128 \times 256 \times 26$), T42 for 2.8 degrees ($64 \times 128 \times 26$), and T31 for 3.75 degrees ($48 \times 96 \times 26$). The 26-level vertical grid uses a hybrid pressure coordinate system. CAM solves the three-dimensional fluid dynamics using the spectral horizontal discretization and fast transform techniques. Physical process models, including radiation transport, convection, moist cloud processes, and precipitation, are largely computed using detailed physical parametrizations (Collins et al., 2004). Time integration is with a centered, three-level (leapfrog) scheme and the semi-implicit system for fast gravity waves solved in the transform domain.

The ocean model is based on the Parallel Ocean Program (POP), version 1.4.3 (Smith and Gent, 2002). It is an ocean circulation model developed at Los Alamos National Laboratory and belongs to a class of ocean models that use depth as the vertical coordinate. POP solves the primitive fluid equations on a sphere using

[1]OAK RIDGE NATIONAL LABORATORY OAK RIDGE, TN 37831-6016, USA (DRAKEJB@ORNL.GOV)

[2]THEORETICAL DIVISION, LOS ALAMOS NATIONAL LABORATORY LOS ALAMOS, NM 87545-1663, USA

[3]NATIONAL CENTER FOR ATMOSPHERIC RESEARCH BOULDER, CO 80307-3000, USA

second-order differencing for the spatial derivatives on a staggered mesh. For climate simulations as part of the CCSM, POP uses a displaced-pole grid (Smith and Kortas, 1995) with the logical North Pole displaced into the Greenland land mass. Such a grid permits simulations of the Arctic regions without the polar singularity. Horizontal resolutions are variable with enhanced resolutions near the equator and in the North Atlantic due to the displaced pole, but typical resolutions are, on average, one degree ($320 \times 384 \times 40$) and three degrees ($100 \times 116 \times 25$). The 40-level vertical grid has variable spacing, starting with 10 m near the surface to better resolve surface and mixed-layer processes and expanding to 250 m in the deep ocean. Solutions are integrated forward in time using a leapfrog time-step for the three-dimensional solution and an implicit preconditioned conjugate gradient solver for the fast barotropic wave modes (Dukowicz and Smith, 1994). Various physical parametrizations, subgrid models, and other features are available (Smith and Gent, 2002).

The land component is the Community Land Model (CLM3). The land model uses a nested subgrid hierarchy of scales representing land units, soil or snow columns, and plant functional types (Bonan et al., 2001; Oleson et al., 2004). The land component operates on the same grid resolution as the atmospheric component.

The sea ice model CSIM5 is based on the Los Alamos CICE model with an elastic–viscous–plastic ice dynamics (Hunke and Dukowicz, 1997, 2002) and the Bitz and Lipscomb thermodynamics (Bitz and Lipscomb, 1999) with multiple ice thickness categories. An incremental remapping scheme (Lipscomb and Hunke, 2004) is used for the transport of ice. The ice model uses the same displaced-pole grid and same resolution as the ocean component (Briegleb et al., 2004).

The precise details of which parametrizations and options are used in the models are described in Collins et al. (2005b).

2 Software Design and Software Engineering

Software engineering is typically defined as a formal process used to design and implement software (Software Engineering Institute, 1995). Little is written on software engineering for codes that develop and evolve over the span of many generations of computer architectures (Drake and Foster, 1995). In the evolution of scientific community codes, such a process must also include collaborative software development and the use of software frameworks. With this broad scope, software engineering and design underlies much of the effort of the CCSM project. Software design of the CCSM is focused on the goals of modularity, extensibility, and portability. These goals must be met at both the component level and within the individual component models.

2.1 MODULARITY AND EXTENSIBILITY

A community research code such as the CCSM must be modular and extensible to enable rapid adoption of new capabilities and new physical parametrizations. Modularity also permits users to choose between many model configurations and to customize the model for specific applications. At the lowest level, the ability to swap, add, or choose between physical parametrizations is necessary and was already a part of most CCSM component models. Use of FORTRAN modules and other software techniques for encapsulation greatly helps maintain modularity at this level. Component models are also modular at the level of major model subcomponents. For example, the CAM and other atmosphere models treat atmospheric dynamics and physics as self-contained subcomponents and are thus able to integrate new dynamical cores and their interactions with the same model physics.

At the model coupling level, it must be possible for alternative component models to be substituted or added. One use of this capability is to provide simplified models for individual components. For example, the CCSM supplies "data" or "dead" models that can be used to mimic components by supplying observed fields or constant fields when a fully interacting component is not necessary. The data-cycling models (data models) are small, simple models that read existing data sets from observations, reanalysis products, or even previous control simulations. These data-cycling components are very inexpensive and are used for both test runs and certain types of model simulations that do not require feedbacks from another component. The dead models are simple codes that facilitate system testing. They generate realistic forcing data internally, require no input data, and can be run on multiple processors to simulate the software behavior of the fully active system.

Swapping components can be used to investigate sensitivity to model formulations. In the third Intergovernmental Panel on Climate Change (IPCC) assessment (Houghton et al., 2001), a coupled climate model with an isopycnal ocean component (density as vertical coordinate) resulted in a different North Atlantic overturning circulation from many of the coupled models with z-coordinate ocean components. The CCSM is currently being used to investigate this issue by comparing a control run using the POP z-coordinate ocean component with an identical run using the isopycnal MICOM (Bleck, 1998) model. At the coupling level, new framework efforts such as the Earth System Modeling Framework (ESMF; Hill et al., 2004) and the Common Component Architecture (CCA; Armstrong et al., 1999;

Bernholdt et al., 2005) are providing more generic component interfaces to the climate and scientific community and should further simplify and enable component interaction.

An important new capability for the CCSM and other climate models is the addition of chemical and biogeochemical interactions. Dynamic vegetation and ecosystem models are being added to the land and ocean components and will interact with atmospheric chemistry models. A design for chemical coupling with the physics modules is partially implemented and is an important test of the CCSM goal of extensibility.

A new dynamical core (Lin and Rood, 1997; Lin, 2004) is being introduced for the atmosphere that is based on a finite volume discretization with a Lagrangian vertical coordinate. This formulation is particularly advantageous for conservative transport of chemical tracers and is receiving much attention. Standard resolutions for the finite volume version of the CAM are 2×2.5 and 1×1.25 degrees, each with 26 vertical levels. The use of the finite volume dynamical core may result in a two-dimensional data and computational decomposition that yields scaling beyond that of the current Eulerian spectral dynamical core (Mirin and Sawyer, 2005).

2.2 PERFORMANCE PORTABILITY

Portability is the ability of a code to compile and run successfully across all platforms. This is largely achieved through adherence to language standards and widely used libraries such as Message Passing Interface (MPI). By "performance portability", we mean the ability of a code to achieve good performance across a variety of computer architectures while maintaining a single source code. As a community model, the CCSM must run on a variety of machine architectures available to the climate community and must perform well on all these architectures for computationally intensive climate simulations. The target machines are most often clusters of commodity cache-based microprocessors. These include Linux clusters, the IBM SP3 and IBM p690. More recently, vector computers, including the NEC SX6 and Cray X1, have become available to the climate community and have added a significant new challenge for performance portability of the CCSM.

There are cases where portability is not possible within a single source code. In these cases, it is the practice to isolate non-portable code into modules or libraries that can be selected at compile time or at run-time. Generic interfaces or wrappers can be defined so that the calling code can remain portable. An example of such a structure is the support of different communications paradigms. Often communication-related routines can be isolated in a small set of routines and called using an interface that is identical whether the underlying code is implemented in MPI, SHMEM, Co-array FORTRAN or copies to a memory buffer. We have avoided use of preprocessor directives (e.g. the C preprocessor *ifdef*) as a primary means of achieving portability; directives often can proliferate and adversely affect code readability as well as complicate testing procedures. Selective use of directives is permitted in cases where code is hidden from the user and well encapsulated.

Climate codes are often limited in performance by memory bandwidth so a key aspect of performance portability is the need to adjust the size and shape of data structures to optimize performance on machines with different cache sizes or with vector processors. Tunable parameters are provided for adjusting data structure size and loop lengths to optimize codes based on problem size, system architecture, and processor configuration. Automated, run-time adjustment of these parameters is a long-term goal, particularly when used to achieve dynamic load balancing. In the short term, compile time parameters are sufficiently easy for scientists to optimize for high processor performance without detailed knowledge of memory bandwidth properties of a particular architecture. This will be described more fully in a later performance section and is the subject of several papers in this special issue (Hoffman et al., 2005; Kerbyson and Jones, 2005; Worley and Drake, 2005).

A key decision made to achieve performance portability was the adoption of a hybrid parallel programming paradigm with independent parallel data decompositions for each component. By not enforcing a single decomposition/communication style, the component developers are free to use the methods that give the best performance. For the atmospheric model, the specification of parallel granularity for both distributed memory structures with parallel MPI communication and shared memory structures with OpenMP parallel task specification are configurable at run-time. The most advantageous settings may vary from machine to machine often with factors of 2 or 3 difference in simulation throughput. The consequence of this design decision is that large amounts of data may be exchanged between components in the course of a simulation. This "data transpose" is localized in a small number of utility routines so that optimization is straightforward, but interconnection bandwidth, latencies, and copy times are limiting factors for scalability and simulation throughput.

2.3 SOFTWARE ENGINEERING PROCESS

A formal software engineering process involves a cycle of defining requirements, designing the software, implementing the design and testing the software (Software Engineering Institute, 1995). Documentation and reviews

at each stage are important to catch bugs early and to avoid costly rewriting when designs do not satisfy current or future requirements. The design documents produced for the CCSM started with descriptions of new scientific requirements of the model. Computational requirements and description of the software architecture led to interface and data structure specification and implementation. Testing was performed at several levels. Unit testing of individual subroutines and modules verified modules work as designed; integrated testing of entire models was used to validate physical fidelity of the model and ensure that all components of a model interact with each other without unintended side effects. Frequent regression tests were required to catch problems generated due to changes in computational environment or bugs introduced during minor maintenance.

Three basic practices support the software engineering process (Hunt and Thomas, 1999): version control, unit testing, and automation. Because the CCSM effort includes a large body of legacy code, the process was adapted to fit the current software. Adopting the entire process was a goal for all new code developed. For existing code, the goal focused primarily on testing and validation. Unit tests and automated testing scripts were developed and used as criteria for acceptance of code changes. The automated test scripts proved very useful to remote developers because they enforced code correctness standards uniformly. A level of confidence on check-in that the simulation capability was not degraded was important for the many ongoing users of the code. The coordination of check-ins was managed by a Change Review Board (CRB) for the atmospheric model and gatekeepers for the ocean, land, and sea ice components. Using the CVS version control system with remote access allowed precise organization and access control of development branches, but also offered risk mitigation when bugs and inefficiencies were inadvertently introduced.

To ease acceptance of new changes and make the process easier for developers, coding standards and testing infrastructure have been created for the community. Coding standards help to encourage encapsulation and modern software practices and also help to maintain a consistent look and feel that users and other developers can understand. A testing infrastructure lowers the burden for developers and encourages good software quality assurance. Information recorded with each check-in included: bit-for-bit reproducibility (implying only structural or performance changes), a list of tests run on which platforms, impact on timing and memory usage, changes to the build system required. Changes producing more than round-off differences in the results were not permitted by a single developer. The CRB required much longer simulations, broader discussions, and scientific review when new modules were introduced which changed the model climate.

This last point should be elaborated and emphasized as it distinguishes scientific software development from business software, and there are profound implications for the software engineering process appropriate to computational science. Due to the mathematical non-linearity inherent in climate system models, it is not possible to anticipate what effect changes in one component will have on the results of other components. Care must be taken to maintain a delicate balance of physical (and biological and chemical) processes within the model in order to guarantee a state-of-the-art, quality simulation product. Changes need to be sequenced, one at a time, so that the relative effects can be tracked and understood. This process of model development and code modification is closely linked with scientific discovery in computational science. Thus, software engineering for climate modeling must involve climate scientists at each step of the process: the specification of requirements, software design, implementation, and testing.

2.4 COLLABORATIVE DEVELOPMENT

The CCSM is an evolving project with over 300 researchers attending the annual CCSM Workshop. While only a subset of these researchers actively contribute to the model, the development team involves a large number of geographically distributed researchers. Tools and processes are required for managing code changes and conflicts. Such tools include common software repositories, version control systems, test requirements, and procedures for introducing code into the repositories. Researchers who choose to invest their efforts in the code framework must be guaranteed the ability to publish their results before code is made available to the wider community. An open source model is therefore not appropriate and repository access has been controlled.

Day-to-day management of the software, repositories and development schedules is the role of the CCSM Software Engineering Group (CSEG) at the NCAR and the component model CRB. Wider input from the community on software issues comes through the Software Engineering Working Group (SEWG). Setting priorities for new scientific capabilities remains in the individual component model working groups, the application working groups and ultimately with the Scientific Steering Committee (SSC). Such a management structure provides for community input and development while maintaining reasonable control of the software quality and routine activities.

Community development will also become easier as the community moves toward componentization and shared utility infrastructure. Important frameworks and

Table 1
Computational performance of CCSM3.0 for IPCC production

Platform	IBM SP3	IBM p690	ES(NEC SX6)	Cray X1
Number of processors	208	192	184	208
Years per day	1.57	3.43	16.2	13.6
Years per day per CPU	0.0075	0.0179	0.0880	0.0654

utilities that are actively being developed include the ESMF (Collins et al., 2005a), the Model Coupling Toolkit (MCT; Larson et al., 2005), the Multiprocessor Handshaking Library (MPH; He and Ding, 2005), and the CCA (Armstrong et al., 1999).

3 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 four 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, 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 a measure of time required to generate the results. The time required in weeks can be large where a thousand years or more of simulation are generated.

3.1 LOAD BALANCING

A primary computational challenge in the CCSM and other climate models is the load imbalance generated by the non-homogeneous structure of a multiphysics, multicomponent model. A striking example of the structure of the load imbalance appears in the calculation of the short-wave radiation balance. This computation need only be done where the sun is shining, i.e. on half 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 imbalance is 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 climate 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 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 and may stall if processor assignments are poorly chosen. As an example, one stage of the sea ice processing is on the critical path to the atmosphere component. The sea ice model must therefore be allocated enough processors to avoid excessive waiting by other components.

Typically, for a fully active T85×1 configuration, two-thirds of the total processor count is assigned to the atmospheric component. The balance of the processors is assigned in part to ensure that the atmospheric processors are kept busy. Balance experiments at T31×3 are shown in Figures 1 and 2 for a variety of computer platforms and processor counts.

3.2 FLEXIBLE DATA DECOMPOSITION

To achieve performance portability, flexible data structures and data decompositions have been introduced in each component model. The ocean and atmosphere models have adopted schemes that rely on chunks or blocks of data that can be tuned for a particular architecture. The chunks or blocks can be sized for either cache or vector machines, providing either cache blocking or long vectors. The chunks or blocks are then distributed in a (currently static) load-balanced manner, depending on estimated work per block. The blocks or chunks can also be oversubscribed to compute nodes to provide a hybrid shared-memory/messaging-passing programming style. The details of these data structures are presented in Worley and Drake (2005), Kerbyson and Jones (2005) and Hoffman et al. (2005). Similar data structures are currently being introduced to the ice model (Lipscomb, private communication). For ocean and sea ice models, such a blocking also provides the capability to remove land or ice-free blocks from the computation, saving computational costs. The land component also utilizes a "clump" scheme to achieve similar results (Hoffman et al., 2005).

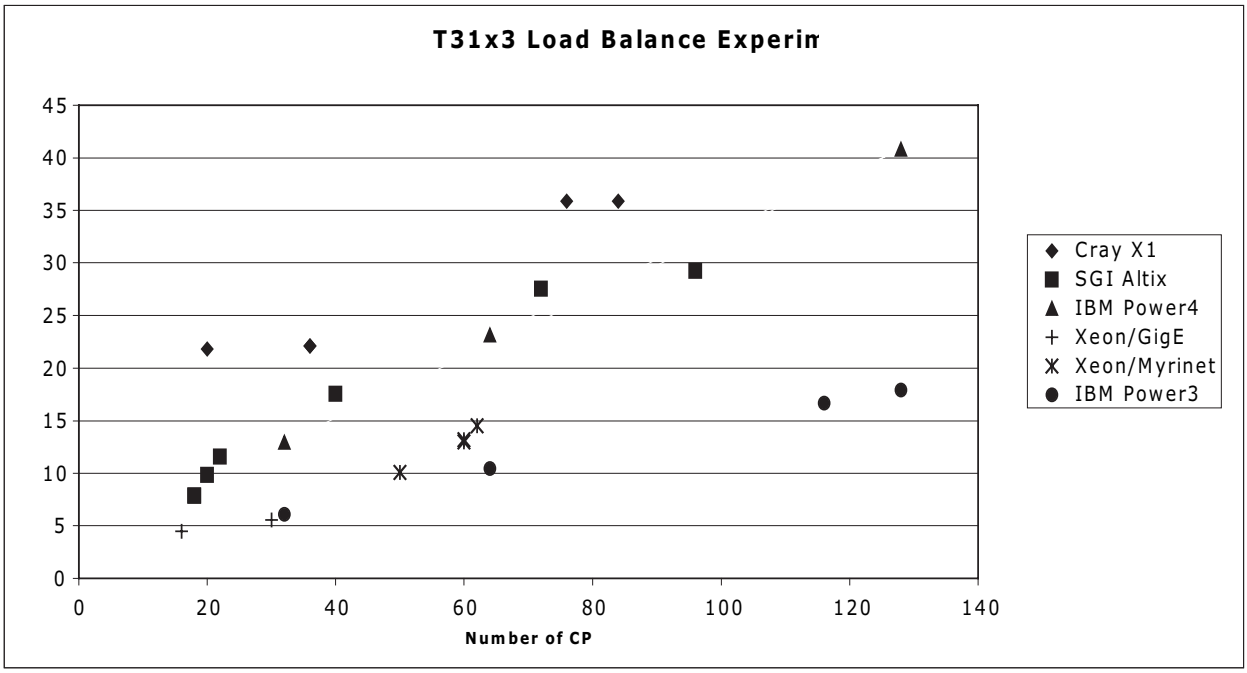

Fig. 1 Simulation throughput in years per day for a number of machines and configurations using the T31×3 resolution. (The numbers plotted are from the initial validation baseline for the Cray and SGI and do not reflect work in progress.)

3.3 VECTORIZATION

Vector computers have once again become available to the climate community with access to both the Cray X1 and the Japanese Earth Simulator (through collaboration with the Central Research Institute for Electric Power Industry (CRIEPI)). The ocean model ported easily to vector architectures as it uses array syntax over horizontal domains, an easily vectorizable construct. However, there were a few routines written for improved cache performance that had to be revised for vector architectures. The atmosphere model evolved from code that had once been vectorized, but required extensive restructuring in the radiation routines (performed by Dave Parks and John Snyder of NEC) and the insertion of many compiler directives for both NEC and Cray machines (performed by Matt Cordery of Cray). Local loop rearrangements in the spectral dynamical core exposed more vectorization to the compilers.

The sea ice and land models were both new models developed with structures and design that were well suited to the science and performed well on cache-based architectures, but prevented vectorization. In the sea ice model thermodynamics, horizontal loops were at a high level with subroutine calls and branching within the loops. In collaboration with CRIEPI (through Clifford Chen of Fujitsu), the ice thermodynamics was restructured, pushing the horizontal loops into subroutines and replacing conditionals within loops with a pre-gather construct. The resulting code vectorizes well and also performs well on cache-based architectures.

Vectorization of the land model required a fundamental rework of the data structures of the CLM (Hoffman et al., 2005).

3.4 COMMUNICATION

Communication affects performance at all levels in the CCSM. Between components, information was initially passed only between master tasks of each component, requiring a gather/scatter process. This has now been

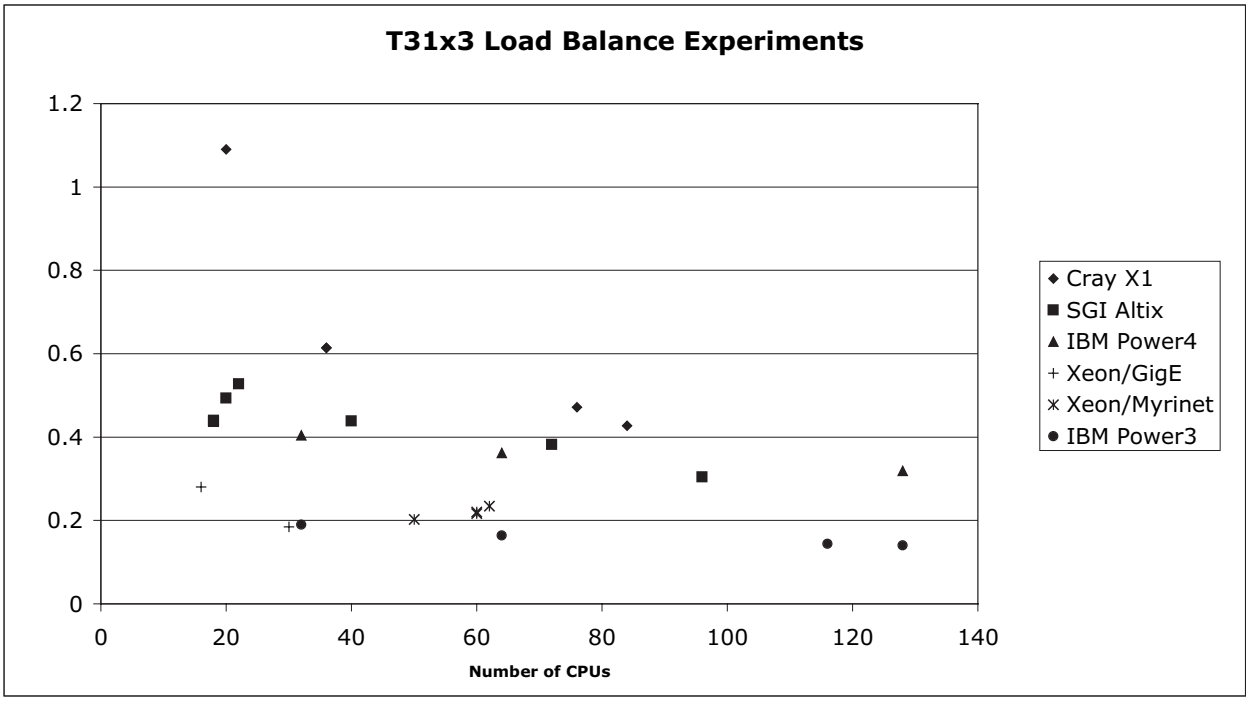

Fig. 2 Efficiency in years per day per CPU for a number of machines and configurations using the T31×3 resolution. Perfect scaling would appear as a horizontal set of points. (The numbers plotted are from the initial validation baseline for the Cray and SGI and do not reflect work in progress.)

replaced by more general all-to-all communication capabilities (Jacob et al., 2005). Communications between the atmospheric model dynamics and physics subcomponents has also been improved and various options can be chosen at run-time to maximize performance for a particular architecture or configuration (Hoffman et al., 2005; Worley and Drake, 2005).

Within components, the use of alternative messaging paradigms is also supported using communication modules and wrappers. This can be particularly useful for improving the scaling of the ocean model's barotropic solver and the halo updates of the finite volume dynamical core for the atmosphere. Performance and scaling of the barotropic solver (a conjugate gradient solver) depends strongly on message latency, so low latency messaging alternatives can result in large performance improvements in the model. On machines with shared memory, the MPI-2 one-sided protocol is advantageous, providing message passing at the cost of a (remote) memory copy.

4 Future software challenges for CCSM

The future scientific direction of the CCSM is largely in the hands of climate researchers who will extend the model to answer specific research questions. The CCSM Science Plan (Kiehl et al., 2004b) calls for extension of capabilities to understand the interaction of aerosols and atmospheric chemistry on climate. The effect of fine-scale ocean eddies on the climate balance and the sources of climate variability are also identified. At the same time, there is a need to improve basic physical aspects of the models that deal with clouds and radiant energy exchanges.

The next few years will see the addition of approximately 100 chemical species to the "transport list" in the atmospheric model along with a consequent increase of computational cost to compute the reactions among these species. Early experiments indicate that the cost per grid point of the atmospheric calculation will go up by a fac-

tor of eight. Similar increases will occur in the ocean due to the addition of ocean biogeochemistry.

Increased resolution to provide regional detail and to resolve significant events, such as hurricanes and typhoons, and to provide better coastal interactions in the ocean model are also generally needed. In an attempt to project the needed computational capabilities, the CCSM Business Plan (Kiehl et al., 2004a) suggests that "a 144-fold increase in computational resources relative to what is currently available" is required to build a "comprehensive system model with appropriate resolution". Over the next ten years we expect the available computers to increase in power from several teraflops to many petaflops. The additional capability will open many fruitful avenues of scientific investigation with coupled Earth system models as a key tool for research. Recent projections for CCSM (Keyes, 2004) covering the next ten years place the computational growth factor at 10^9–10^{12} in order to address the scientific questions of climate science.

The role of dynamic vegetation (Levis et al., 2004) in the carbon cycle and the feedbacks with climate change relate to a complex set of biological and chemical processes that must be modeled and simulated to quantify their interactions. New observational capabilities from satellites are allowing the construction of detailed land use and ecological characterizations. These provide a challenge to modelers to use as driving boundary conditions for historical simulations of climate and to develop process models that duplicate observed behaviors.

Application of the model to global climate change studies and to evaluate future emission scenarios also challenges the computational capabilities of present centers and available supercomputers. As a case study, the simulations supporting the United States' contribution to the IPCC 2007 report (Watson et al., 2001) have produced approximately 10,000 years of climate simulations with over 100 Tbytes of simulation output. The resource required using the present model for these computations is 27.5×10^6 CPU-hours of an IBM SP3 (NHII). The production simulation phase of the project ended in late 2004 and used resources provided by NCAR (IBM p690), Oak Ridge National Laboratory (IBM p690), National Energy Research Scientific Computing Center (IBM SP3) and the Japanese Earth Simulator (NEC SX6). Each future scenario involved an ensemble of one to five computational experiments in order to characterize the dynamic uncertainties and confidence intervals of the model forecast.

To effectively utilize the high-end computing capabilities of modern supercomputers, developers of coupled climate models face two continuing challenges: scalability and load imbalance. As longer simulations are required, the spatial resolution must be lowered. Because most scalable parallel algorithms are based on a domain decomposition technique that splits the data (and processing) across the nodes of the machine, there is a limited amount of parallelism due to low resolution. Compounding this effect is the limitation on time-step size imposed by a stability criteria based on the minimum mesh spacing. As the resolution is increased the number of time-steps to complete a simulation also increases, which, in turn, increases the computational cost of the simulation. Since the costs associated with time stepping are inherently sequential, the National Research Council report *Improving the Effectiveness of U.S. Climate Modeling* (Sarachik, 2001) notes that Ahmdal's law applies. With serial portions of the algorithm at 0.1%, Amdahl's law implies the maximum (asymptotic) speedup is 1000. The speed of the computer on this serial portion of the code is highly relevant for climate simulation throughput.

The cost of memory access and communication in a distributed memory parallel algorithm is also critical for scalability of the climate codes during a given time-step. Ordering of the indices and distribution of the data structures for effective memory access in one phase of the calculation (e.g. fast Fourier transforms) may be different from another phase, thus requiring a data transposition and introducing a dependency on the interconnect bandwidth and latency. This effect is clearly evident when comparing the performance of the CCSM on machines with fast interconnects. The memory access pattern for significant portions of the atmospheric physics code shows ratios of the number of floating point operations to memory access in the range of 1 or 2. These portions show good performance on vector processors such as the NEC SX6 or the Cray X1, and must be fit into cache for good performance on scalar processors. Because of the sensitivity of the data transpose operations to memory bandwidth, the nearest-neighbor halo updates to latency, the physics calculations to memory and processor speed, it is evident that balanced machine architectures offer an advantage for climate simulation codes.

Several papers in this issue describe steps taken to reduce the load imbalance and to address the issues of scalability. In the coupled system, one "bad component" can overshadow the good performance of all the other components, so it is important to hold component development to a high standard and not allow the simulation capabilities to be degraded. This can be accomplished by continued attention to the software engineering process and the design of the code for an open community of developers.

ACKNOWLEDGMENTS

This work was prepared under the auspices of the Oak Ridge National Laboratory managed by UT-Battelle for the DOE under contract DE-AC05-00OR22725, and by

Los Alamos National Laboratory managed by the University of California under W-7405-ENG-36 and by the NCAR sponsored by the National Science Foundation. We gratefully acknowledge support from the DOE Office of Biological and Environmental Research as well as the National Science Foundation. Special thanks to Dr. Robert Malone, who started this endeavor, Dr. William Collins, the Chief Scientist for the CCSM project, and to Dr. Warren Washington who manages the Climate Change Research Section at NCAR. The coordination of the CCSM activity has greatly benefited from the efforts of Tony Craig, Tom Henderson, and Mariana Vertenstein.

AUTHOR BIOGRAPHIES

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.

Phil Jones is the project leader of the Climate, Ocean, and Sea Ice Modeling Project within the Theoretical Fluid Dynamics Group at Los Alamos National Laboratory. His research interests include coupled climate modeling, ocean modeling, remapping and interpolation and computational performance of climate models. He holds a Ph.D. in astrophysical, planetary, and atmospheric science from the University of Colorado and a B.S. in physics and mathematics from Iowa State University.

George R. Carr, Jr. received a B.A. in mathematics from Drake University in Des Moines, Iowa and an M.S. in computer science from Ohio State University. He has spent his entire career working in high performance computing and is currently working at the NCAR on portability and performance of the CCSM.

References

Armstrong, R., Gannon, D., Geist, A., Keahey, K., Kohn, S., McInnes, L., Parker, S., and Smolinski, B. 1999. Toward a common component architecture for high performance scientific computing. *Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing*, Redondo Beach, CA, August 3–6.

Bernholdt, D. E. et al. 2005. A component architecture for high performance scientific computing. *International Journal of High Performance Computing Applications* 19(3).

Bleck. R. 1998. Ocean modeling in isopycnic coordinates. Ocean Modeling and Parametrization, E. P. Chassignet and J. Verron, editors, *NATO Science Series*, Kluwer, Dordrecht, pp. 423–448.

Bitz, C. M. and Lipscomb, W. H. 1999. An energy-conserving thermodynamics model of sea ice. *Journal of Geophysical Research* 104:15,669–15,677.

Bonan, G. B., Levis, S., Kergoat, L., and Oleson, K. W. 2001. Landscapes as patches of plant functional types: an integrating approach for climate and ecosystem models. *Global Biogeochemical Cycles* 16, 5.1–5.23.

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

Collins, W. D. et al. 2004. Description of the NCAR Community Atmosphere Model (CAM 3.0). NCAR Technical Note NCAR/TN-464+STR, p. 226.

Collins, N., Theurich, G., DeLuca, C., Trayaonv, A., Li, P., Yang, W., and Hill, C. 2005a. Design and implementation of Earth System Modeling Framework components. *International Journal of High Performance Computing Applications* 19(3).

Collins, W. D. et al. 2005b. The Community Climate System Model: CCSM3. *Journal of Climate*. In press.

Drake, J, and Foster, I. 1995. Special Issue on Climate and Weather Modeling. *Parallel Computing* 21(10).

Dukowicz, J. K. and Smith, R. D. 1994. Implicit free-surface method for the Bryan–Cox–Semtner ocean model. *Journal of Geophysical Research* 99:7991–8014.

He, Y. and Ding, C. 2005. Coupling multicomponent models by MPH on distributed memory computer architectures. *International Journal of High Performance Computing Applications* 19(3).

Hill, C., DeLuca, C., Balaji, V., Suarez, M., DaSilva, A., and ESMF Joint Specification Team. 2004. The architecture of the Earth System Modeling Framework. *Computing in Science and Engineering* 6:21–28.

Hoffman, F., Vertenstein, M., Kitabata, H., and White, T. 2005. Vectorizing the Community Land Model (CLM). *International Journal of High Performance Computing Applications* 19(3).

Houghton, J., Ding, Y., Griggs, D., Noguer, M., van der Linden, P., and Xiacsu, D. 2001. Contribution of Working Group I to the Third Assessment Report of the Intergovernmental Panel on Climate Change, Cambridge University Press, Cambridge, p. 944.

Hunke, E. C. and Dukowicz, J. K. 1997. An elastic–viscous–plastic model for sea ice dynamics. *Journal of Physical Oceanography* 27:1849–1867.

Hunke, E. C. and Dukowicz, J. K. 2002. The elastic–viscous–plastic sea ice dynamics model in general orthogonal curvilinear coordinates on a sphere – incorporation of metric terms. *Monthly Weather Review* 130:1848–1865.

Hunt, A. and Thomas, D. 1999. The Pragmatic Programmer: From Journeyman to Master, Addison-Wesley, Reading, MA.

Jacob, R., Larson, J., and Ong, E. 2005. M × N communication and parallel interpolation in CCSM3 using the Model

Coupling Toolkit. *International Journal of High Performance Computing Applications* 19(3).
Kerbyson, D. J. and Jones, P. W. 2005. A performance model of the Parallel Ocean Program. *International Journal of High Performance Computing Applications* 19(3).
Keyes, D. 2004. A Science-based Case for Large-scale Simulation. Office of Science, U.S. DOE, Vol. 2, p. 74 (available from http://www.siam.org/siamnews/09-03/SCaLeS.htm).
Kiehl, J., Hack, J., Gent, P., Large, W., Blackmon, M., Bretherton, C., Chang, P., Bitz, C., Doney, S., and McKenna, D. 2004a. The CCSM Business Plan for 2004–2008 (available from http://www.ccsm.ucar.edu).
Kiehl, J. et al., editors. 2004b. The CCSM Science Plan for 2004–2008 (available from http://www.ccsm.ucar.edu).
Larson, J., Jacob, R., and Ong, E. 2005. The Model Coupling Toolkit: a new FORTRAN90 toolkit for building multiphysics parallel coupled models. *International Journal of High Performance Computing Applications* 19(3).
Levis, S., Bonan, G. B., Vertenstein, M., and Oleson, K. W. 2004. The Community Land Model's Dynamic Global Vegetation Model (CLM-DGVM): Technical Description and User's Guide. NCAR Technical Note NCAR/TN-459+IA, p. 50.
Lin, S. J. 2004. A vertically Lagrangian finite-volume dynamical core for global models. *Monthly Weather Review* 132(10):2293–2307.
Lin, S. J. and Rood, R. 1997. An explicit flux-form semi-Lagrangian shallow-water model on the sphere. *Quarterly Journal of the Royal Meteorological Society* 123:2477–2498.
Lipscomb, W. and Hunke, E. C. 2004. Modeling sea ice transport using incremental remapping. *Monthly Weather Review* 132:1341–1354.
Mirin, A. and Sawyer, W. 2005. A scalable implementation of a finite-volume dynamical core in the Community Atmosphere Model. *International Journal of High Performance Computing Applications* 19(3).
Oleson, K. W. et al. 2004. Technical description of the Community Land Model (CLM). NCAR Technical Note NCAR/TN-461+STR.
Sarachik, E. 2001. Improving the Effectiveness of U.S. Climate Modeling. National Academies Press, Washington, DC.
Smith, R. D. and Gent, P. 2002. Reference manual for the Parallel Ocean Program (POP). Los Alamos Unclassified Report LA-UR-02-2484.
Smith, R. D. and Kortas, S. 1995. Curvilinear coordinates for global ocean models. Los Alamos Unclassified Report LA-UR-95-1146.
Software Engineering Institute at Carnegie Mellon University. 1995. The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, Reading, MA.
Washington, W. M., 1982. Documentation for the Community Climate Model (CCM) Version 0. NCAR report, Boulder Colorado, NTIS No. PB82 194192.
Watson, R. T. and Core Writing Team. 2001. IPCC Third Assessment Report: Climate Change 2001. IPCC, Geneva.
Williamson, D. L. 1983. Description of the NCAR Community Climate Model (CCM0B). NCAR Technical Note NCAR/TN-210+STR, Boulder, CO, NTIS No. PB83 231068.
Worley, P. H. and Drake, J. B. 2005. Performance portability in the physical parametrizations of the Community Atmospheric Model. *International Journal of High Performance Computing Applications* 19(3).