DEVELOPMENT OF A MODULAR, PERFORMANCE-PORTABLE CLIMATE SYSTEM MODEL

Final Report of the ACPI *Avant Garde* Project

For the period June 1, 2000, to September 30, 2001

Submitted to DOE Office of Science
November 21, 2001

Authors

Thomas Bettge	National Center for Atmospheric Research Climate and Global Dynamics Division	bettge@ucar.edu
Maurice Blackmon	National Center for Atmospheric Research Climate and Global Dynamics Division	blackmon@ucar.edu
Byron Boville	National Center for Atmospheric Research Climate and Global Dynamics Division	boville@ucar.edu
Anthony Craig	National Center for Atmospheric Research Climate and Global Dynamics Division	tcraig@ucar.edu
Cecelia Deluca	National Center for Atmospheric Research Scientific Computing Division	cdeluca@ucar.edu
Chris Ding	Lawrence Berkeley National Laboratory	chqding@lbl.gov
John Drake	Oak Ridge National Laboratory	bbd@msr.epm.ornl.gov
Peter Eltgroth	Lawrence Livermore National Laboratory	eltgroth1@llnl.gov
Ian Foster	Argonne National Laboratory	foster@mcs.anl.gov
Robert Jacob	Argonne National Laboratory	jacob@mcs.anl.gov
Phil Jones	Los Alamos National Laboratory	pwjones@lanl.gov
Brian Kauffman	National Center for Atmospheric Research Climate and Global Dynamics Division	kauff@cgd.ucar.edu
Jay Larson	Argonne National Laboratory	larson@mcs.anl.gov
Arthur Mirin	Lawrence Livermore National Laboratory	mirin@llnl.gov
Everest Ong	Argonne National Laboratory	etong@mcs.anl.gov
Doug Rotman	Lawrence Livermore National Laboratory	rotman1@llnl.gov
David Williamson	National Center for Atmospheric Research Climate and Global Dynamics Division	wmson@ucar.edu
Patrick Worley	Oak Ridge National Laboratory	worleyph@ornl.gov

Contents

Executive Summary

The Climate System Model (CSM) and Parallel Climate Model (PCM) of the National Center for Atmospheric Research (NCAR) are two advanced climate models that have seen significant use in climate research and studies of climate variability and global climate change. The community CSM-1 model links atmospheric, oceanic, biologic, cryogenic, and chemical components; it has been and continues to be used for a wide range of climate research. Developed with DOE support, PCM-1 couples similar models and, in addition, has been adapted to execute on scalable parallel computers, hence allowing long-duration simulations in support of DOE missions.

Recognizing the strengths of these two models, NCAR scientists began merging the CSM and PCM code bases to produce the Community Climate System Model, CCSM, with the goal of achieving significant improvements in model performance. As they tackled this goal, NCAR staff faced two significant challenges. First, CSM was not designed to exploit the microprocessor-based scalable parallel-architecture computers that are currently being deployed at NSF and DOE centers. A consequence of this limitation is that performance has not increased substantially in the past five years. Second, both CSM and PCM model structures needed to be improved with a view to enabling "plug and play" substitution of important modules, such as dynamical solvers and physics packages. This latter improvement would both facilitate ongoing development of the new merged CSM-2 model and make it easier for scientists to experiment with improvements to individual components.

A group of DOE and NCAR scientists thus proposed a joint R&D activity aimed at developing a next-generation modular, performance-portable CCSM. This work was expected to produce two primary outcomes: a performance-enhanced CCSM, better able to exploit microprocessor-based parallel computers, and a detailed design for current and future CCSM versions with substantial improvement in terms of modularity and portability. A substantial challenge in both areas was to evolve software engineering practices without unduly disrupting CCSM development or diverging from a common code base.

The R&D activity tackled, in particular, the design and development of (1) a scalable, modular atmosphere model and (2) a next-generation coupler. In the atmosphere domain, work focused on improving node performance, developing a more modular atmosphere model structure that permits the substitution of both dynamics and physics components, and developing the high-performance communication libraries required for good performance on scalable parallel computers. Work on the coupler addressed issues of scalability and configurability. Work on scalability is important because the CSM coupler did not feature distributed-memory parallelism, and the CSM did not scale beyond around 64 processors. Work on configurability is important because users want to be able to use CSM components in a wide variety of modes more easily than with the current coupler. The DOE/NCAR also worked on improving ocean model and I/O performance on parallel computers.

The proposal specified concrete, realizable tasks and milestones for both DOE and NCAR participants. These activities supported the scientific directions for CCSM, as defined by the CCSM Scientific Steering Committee, and also the goals of DOE's CCPP Program. Day-to-day activities were coordinated by a small management group, which worked closely with the recently formed CCSM Software Engineering Working Group. Participants in the proposed project were active participants in that working group. A significant auxiliary outcome of this project was the development and successful validation of the techniques and working relationships required to enable productive collaborative development by a multilaboratory and multi-agency team.

This final report summarizes the results obtained in this 18-month project, which was completed on September 31, 2001. During this period, the goals of the project were largely achieved. In particular, the project produced detailed software design documents for two CCSM components; a major restructuring of the atmospheric model into modules; a scalable, modular coupler toolkit and coupler system; performance-portable parallel implementations of atmosphere model; and improvement of the atmospheric dynamical cores. In addition, the process of code development was changed to support modern software engineering practices of a distributed, multi-agency team by the establishment of a community code repository for CCSM developers; defining developer guidelines; deploying extensive, multimachine testing and validation procedures; and automating developer notification of updates and project coordination information. In the past 18 months the way of doing business in the CCSM has evolved in response to and because of the Avant Garde project.

1 Introduction

The NCAR Climate System Model (CSM) project started in January 1994 and led to the 1996 release of CSM version 1.0 (CSM-1), which coupled atmosphere, ocean, land surface, and ice model components. Various aspects of the model are described in a special issue of the *Journal of Climate* (Climate System Model 1998). Concurrent with this activity, a group led by Warren Washington at NCAR developed the Parallel Climate Model (PCM) with the goal of enabling long-duration climate simulations on scalable parallel computers (Washington et al. 2000). A merging of these two model development activities was proposed in mid 2000, with the goal of producing a substantially new community climate system model, CCSM, within a year.

The developers of any major scientific simulation system—and in particular the developers of a system as complex as CCSM—face two major challenges: performance portability and extensibility.

Performance portability arises as a challenge because of a simultaneous increase both in performance requirements (e.g., to support long-duration, high-resolution climate simulators, and ensemble studies) and in the range of computer architectures in use today. In addition to traditional vector machines, codes such as CCSM must be able to operate efficiently on microprocessor-based distributed-memory computers, shared-memory computers, and hybrid systems. On these systems, memory hierarchies are complex, and memory bandwidth issues dominate performance. Future systems can only be expected to become more challenging in this regard (Sterling et al. 1995). Historically, NCAR models have emphasized vector performance and modest parallelism. New approaches to model development are required if we are to construct models that achieve good performance on a range of platforms while remaining usable by the scientists who must develop and maintain them. Successful projects at the European Center for Medium-Range Weather Forecasting and at Argonne National Laboratory in collaboration with NCAR's Mesoscale and Microscale Meteorology Division (Michalakes 1998, 2000, 2001) suggest that this goal can be achieved: but significant challenges remain, particularly in the context of more complex coupled models.

Extensibility arises as a challenge because the community nature of CCSM means that a large number of developers need to experiment with modifications and extensions to the core CCSM framework. For example, scientists located at geographically distant sites may incorporate advanced submodels, alternative dynamical cores, new physical parameterizations, and climate processes such as clouds, chemical interactions, and surface and subsurface water transport. If these modifications and extensions are not easy to perform, the utility of the model as an experimental framework is significantly reduced, the productivity of the CSM model development is curtailed, and the long-term scientific viability of CSM is compromised. Yet while NCAR models are well engineered on the whole, they have not been designed with a view to extensibility, modularity, and collaborative development. (For example, there was no overall "CCSM design document" when this project began.) Again, experiences elsewhere in the scientific computing community suggest that modularity can be achieved, even in the challenging case of high-performance scientific codes (Armstrong et al. 1999); but achieving this goal in the context of CCSM requires significant effort and commitment.

These considerations suggested that the utility of the CCSM effort could be enhanced significantly by a focused attack on these two challenges. Furthermore, the fact that the CSM and PCM development teams at NCAR had committed to the development of CCSM means that such an activity would be particularly timely, making it possible to produce a CCSM that was significantly enhanced in these two areas. To this end, we proposed in early 2000 an R&D project (entitled "Avant Garde" as it was intended as a precursor to the Accelerated Climate Prediction Initiative) with two main goals: first, to restructure key CSM components with a view to enhancing performance on a range of platforms, including scalable microprocessor-based parallel computers; and second, to develop the design principles and documents that can guide future CCSM development with a view to enhancing performance and extensibility. This project was funded in May 2000 as a partnership between DOE laboratory scientists and software engineers, who provided expertise in parallel computing and software engineering, and NCAR scientists and software engineers, who provided expertise in computational physics as well as parallel computing and software development.

The rest of this report motivates the approach taken in this project, discusses the technical results that were achieved, discusses the organizational structures that we used to achieve success, and makes recommendations for future work in this area.

Section 2 describes CSM and PCM, their principal components, and the performance characteristics of those components in their current form. The information in this section provides background information that informs subsequent technical discussion and also makes clear the nature of the performance challenges facing the developers of CSM-2.

The goal of performance portability implies that we wish to produce a model that can execute efficiently on a range of platforms and in a variety of configurations. To focus our efforts, we proposed a small number of target platforms and resolutions, selected with a view both to meeting immediate NCAR and DOE needs and to covering the space of interesting configurations. These target platforms and resolutions are described in Section 3.

Section 4 describes the technical work undertaken in this project. We focused on two primary CSM-2 components: the atmosphere model and coupler. Work on these two components, plus supporting work on performance tuning, parallel I/O, and ocean model performance, which are now being integrated to create a performance-enhanced CSM-2 capable of achieving higher performance than the current CSM via both more efficient use of individual processors and efficient execution on larger numbers of processors. An important side product of this work the stimulus and contribution to the development of a detailed model design for CSM-2 as a whole. This model design elucidates primary model components and interfaces and identifies coding standards and testing procedures designed to enhance extensibility.

A particular challenge that we faced in designing and executing this project was engaging CSM-2 developers in an "open software design process" while simultaneously maintaining the productivity of NCAR scientists testing and using CSM-2. Only a well-planned and effectively executed project can build a long-term, sustainable model development collaboration and ensure productive joint work by multiple laboratories and multiple agencies. Hence, Sections 5 and 6 describe the design processes and management structure, respectively, that we adopted to facilitate collaborative model development.

2 NCAR's Climate System Model

By way of background we review the status of the NCAR CSM-1 and PCM at the beginning of this project.

2.1 Overview

The effort to develop and support a community model for climate studies began 20 years ago. Washington (1982) described the first community atmospheric model CCM0A. This model was followed by CCM0B in 1983. The second-generation community model, CCM-1, was introduced in 1987, and included a number of significant changes to the model formulation, which were manifested in changes to the simulated climate. The third generation of the CCM, CCM-2, was released in 1992 and improved the physical representation of key climate processes, including clouds and radiation moist convection, the planetary boundary layer, and large-scale transport. The introduction of this model also marked a new philosophy with respect to implementation. The CCM-2 code was entirely restructured so as to satisfy three major objectives: greater ease of use, including portability across a range of computational platforms; conformance to a plug-compatible physics interface standard; and the incorporation of single-job multitasking capabilities. A steady improvement in the simulated climate of the CCMs is well documented, along with more extensive treatment of physical processes.

An atmospheric climate model alone is not suitable for long-range studies of climate and climate variability. An active ocean general circulation model must be coupled with the atmospheric model even to simulate the important climate variations. The ENSO is an example of a coupling between the ocean and atmosphere that occurs on the interannual scale. In the late 1980s, as a result of advances in the scientific understanding of these interactions and the increased power of computing, the first community coupled models were developed. The NCAR CSM and PCM models are two of the premier next-generation efforts.

Coupled atmosphere and ocean general circulation models (GCMs) had become commonly used in the late 1990s for studies of the natural variability of the climate system and its response to changes in greenhouse gases and aerosol radiative forcings. The NCAR Climate System Model, version one CSM-1 is a physical climate model similar in nature to several other coupled models that have been used for climate studies (see Gates et al. 1996 and Kattenberg et al. 1996). The main features in CSM-1 compared with other coupled climate models are the coupling strategy and state-of-the-art parameterizations, especially in the ocean model.

CSM-1 contains an atmospheric GCM, an oceanic GCM, a land surface biophysics and basic soil hydrology model, and a sea-ice dynamics and thermodynamics model. These component models communicate through a driver program called the flux coupler, which controls the time coordination of the integration and calculates most of the fluxes at the interfaces between the model components. The philosophy has been adopted in the CSM that the most appropriate boundary conditions for the component models are the fluxes at the earth's surface. Those interfacial fluxes that depend directly on the state of more than one component model – for example, turbulent fluxes of latent and sensible heat – are computed within the flux coupler. No flux corrections in momentum, heat, or freshwater are applied. The flux coupler is also responsible for interpolating and averaging between the different grids of the component models while conserving local and integral properties. The surface atmospheric fields are interpolated to the finer grid of the ocean model, and the fluxes are calculated on the ocean model grid. The fluxes are then averaged back onto the coarser atmospheric model grid. This interaction becomes increasingly important if the ocean model has much higher resolution than the atmospheric model, because the higher resolution information affects the local turbulent fluxes.

The flux coupler currently allows two separate coupling intervals between itself and the component models. The atmosphere, land, and sea-ice models communicate at the faster interval, usually one hour, and the ocean model communicates at the slower interval, usually one day. Instantaneous values of state variables and interfacial fluxes time averaged over the coupling interval are passed. Therefore, fluxes are computed from instantaneous state variables, and the time integrals of the fluxes applied in the different model components are the same.

The coupling strategy allows component models to be interchanged relatively easily. Each component model is isolated from the others and from the coupler, across a predefined message-passing interface. Therefore, different models can be used for any component without affecting the rest of the modeling system. For example, the ocean model can be a simple program to supply specified sea surface temperatures, or it can be the full ocean GCM. A tropical Pacific upper-ocean model can also be used for

seasonal to interannual simulations. Similarly, the atmospheric component can be either CCM-3 or a program supplying results of previous simulations or atmospheric analyses. This flexibility is exploited during the spinup phase. The execution of the component models can even be distributed across different computers, a feature that has been demonstrated but is rarely used.

The Parallel Climate Model is a coupled model with many of the same components as CSM-1. It has played an important role in DOE-sponsored historical (**Figure 1**) and enhanced greenhouse-gas scenario simulations, and considerable effort has been invested to achieve good parallel performance. The flux-coupling strategy of PCM differs from that of CSM-1 in the way the land surface model is coupled with the atmosphere and in the manner in which component model execution is synchronized. The land surface and atmospheric coupling are built into the atmospheric component. The component models are configured as part of a single computer program, single executable, with components called sequentially. This strategy precludes the execution of component models across different computers; flexibility has been sacrificed in order to increase operational performance on target platforms.

The PCM and CSM ocean model components are based on different ocean models, Parallel Ocean Program for PCM and NCOM for CSM-1. The sea-ice models are also different, with PCM using Zhang and Hibler (1997) sea ice dynamics with line relaxation for solving the viscous-plastic ice rheology (the rheology of Hunke and Dukowicz 1997 is also supported as an option) and CSM-1 using the rheology of Flato and Hibler (1992).

The work undertaken in the Avant Garde project supports the established directions of NCAR scientists and CSM management. By enabling performance portability for CCSM development, we make possible coupled climate simulations with adequate resolution to simulate weather systems, ocean eddies, and surface exchange processes that affect climate dynamics. By creating an extensible design for CCSM development, we allow the incorporation of advanced submodels, parameterizations, and climate processes that will continue and accelerate the progress toward more comprehensive models that simulate climate with higher accuracy and fidelity.

2.2 Atmosphere Model

The atmospheric GCM incorporated in both CSM and in PCM is CCM-3, which is described in Kiehl et al. (1998a, 1998b), Hack et al. (1998), Hurrell et al. (1998) and Briegleb and Bromwich (1998). CCM-3 is the latest generation of the Community Climate Model from NCAR, with several major improvements over the previous version (CCM-2), primarily in the parameterization of hydrologic processes and in the radiative properties of clouds. CCM-3 is a spectral model; and the standard configuration, documented in the above papers, employs T42 truncation (~ 2.8 degrees) with 18 levels in the vertical. Penetrative convection is parameterized by the scheme of Zhang and McFarlane (1995), and the scheme of Hack (1994) is used for shallow convection. Cloud fractions and optical properties are computed diagnostically from large-scale variables and convective mass fluxes (Kiehl et al. 1998a). The nonlocal boundary layer turbulent flux parameterization is an updated version of Holtslag and Boville (1993), giving lower boundary layer depths and higher surface humidities. The long-wave radiation treats the effects of CO_2, O_3, H_2O, CH_4, N_2O, CFC11, and CFC12. With specified present-day sea surface temperatures, CCM-3 produces a globally and annually averaged balance between incoming solar radiation and outgoing long-wave radiation to less than 0.5 W/m^2.

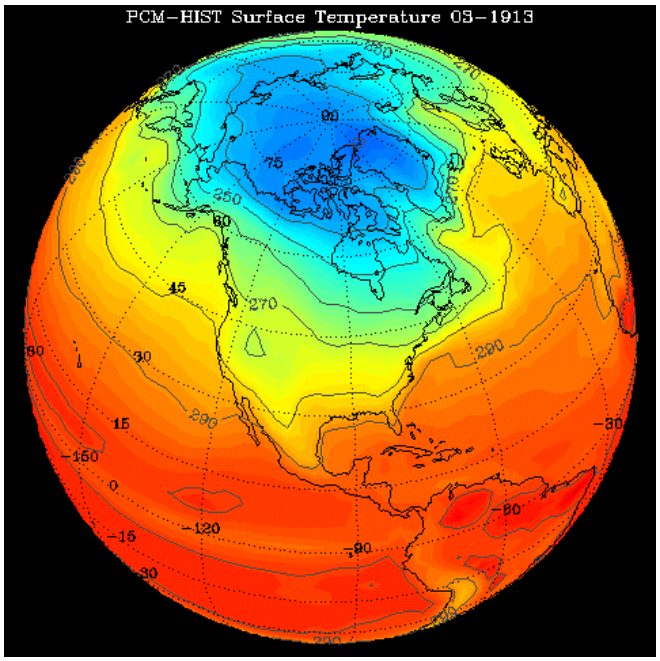

Figure 1: Surface temperature produced by CCM-3 in a PCM historical simulation

A 1D parallel decomposition is implemented in CCM-3, which provides for up to 64 parallel tasks in the grid point space and 43 parallel tasks in the spectral space at the standard T42 horizontal resolution. The decomposition by latitude has been used successfully on moderately sized systems, and for the past several years most of the CSM and PCM simulations have been performed with configurations of 32 or 64 processors. Studies of performance based on a more general 2D parallel decomposition of CCM-2 (Drake et al. 1995) and CCM-3 indicate that the longitude direction should also be decomposed when using more than 32 processors. The optimal aspect ratio depends on machine characteristics; on the NERSC Cray T3E-900, 512 processors can best be used with a 16(lon)x32(lat) decomposition. For T42 resolution, parallel efficiency begins to decline after 32 processors, with performance per processor dropping by half at 512 processors.

2.3 Ocean Model

The ocean component planned for CSM-2 (and used in the current PCM) is the Parallel Ocean Program (POP) from Los Alamos National Laboratory. POP is a descendant of the Bryan-Cox-Semtner (BCS) class of z-level ocean models and was developed under the sponsorship of the Department of Energy's CHAMMP program. A number of improvements were developed and incorporated in POP both for performance on parallel computers and for improved ocean simulations. Significant algorithmic improvements over previous BCS models include a surface pressure formulation and implicit solution of the barotropic mode, a free surface boundary condition, and pressure averaging. These improvements permit longer time steps and allow the use of unsmoothed, realistic topography. Details of the model are found in articles by Smith et al. (1992), Dukowicz et al. (1993), and Dukowicz and Smith (1994). POP also supports general orthogonal grids in the horizontal coordinates. In particular, the use of displaced-pole grids (see **Figure 2**, Smith et al. 1995) in which the pole in the northern hemisphere is displaced into land masses permits accurate solutions of the Arctic regions without filtering or severe time step restrictions related to convergence of grid lines. A number of improved physical parameterizations have recently been added to POP. These are the Gent-McWilliams (1990) isopycnal mixing scheme, the KPP vertical mixing scheme (Large et al. 1994), and an anisotropic horizontal viscosity. The last parameterization gives improved equatorial currents at coarse resolution but is not necessary at higher resolutions, such as 2/3°.

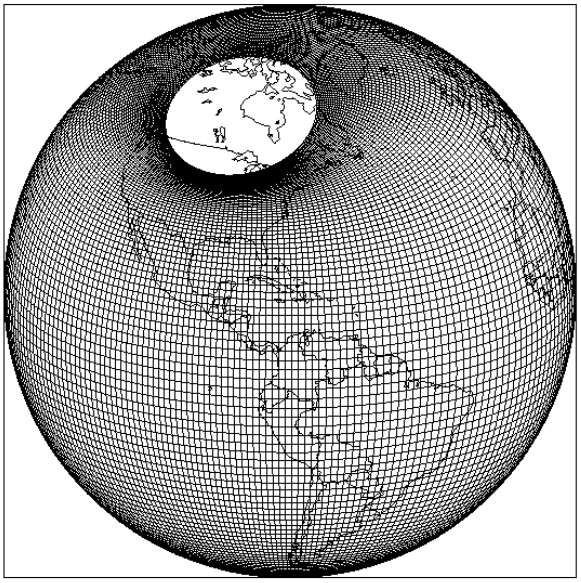

Figure 2: A displaced-pole grid

POP was designed from the beginning to run on massively parallel computers and has continued to evolve as machine architectures have evolved. It is designed to be very portable and runs on most available machines with *no* changes in the source code and *no* architecture-specific preprocessor options. Interprocessor communication details are encapsulated in a very few modules that support MPI, Cray SHMEM, and serial execution, with the choice of communication paradigm being determined by specifying the appropriate directory during the build process. Work is under way on a hybrid MPI/OpenMP implementation that will allow more efficient use of clusters of SMP boxes.

2.4 Land Surface, Sea-Ice, and River Transport Model

The NCAR Land Surface Model (Bonan 1998) simulates the biogeophysical and biogeochemical land-atmosphere interactions, especially the effects of land surfaces on climate and atmospheric chemistry. The LSM runs on the same grid as CCM-3; but rather than attempting to define an average land and vegetation type for each grid cell, the cells are subdivided into four different surfaces, allowing differing vegetation type, bare soil, lakes, and wetlands to be treated separately. Grid cell average fluxes are determined by fractional area-weighted merging of the fluxes of each surface type. A river runoff model was included that balances freshwater in the CSM. The LSM surface fluxes are tightly coupled with the atmospheric

GCM but operate as a separate executable in the CSM. Currently, surface points containing land are statically partitioned to processors, with MPI-style communications flowing through the flux coupler.

The sea-ice component of PCM is based on the CICE framework developed at Los Alamos National Laboratory by Hunke and Lipscomb (1999). This model contains three interacting components: (1) a thermodynamic model based on Bitz and Lipscomb (1999) produces local growth rates of snow and ice due to vertical conductive fluxes, snowfall, and local temperatures; (2) a model of ice dynamics predicts the velocity field of the ice pack based on a model of the material strength of the ice; and (3) a transport model describes advection of the areal concentration, ice thickness, and snow depths. CICE uses an elastic-viscous-plastic rheology for improved ice dynamics and an improved ice advection algorithm. The model will incorporate multiple ice thickness categories, initially following the work of Bitz et al. (2000). CICE is fully explicit in its time integration and is implemented by using both MPI and OpenMP parallelism. Results show that the CICE model scales well for small numbers of processors (5 to 10 processors); but because sea ice is essentially a two-dimensional phenomenon at the ocean/atmosphere boundary, scaling may be limited at high processor counts where the surface-to-volume ratio is large.

2.5 Coupler

The coupler is the model component responsible for coordinating the execution of the various components that form a coupled model. It serves four primary functions (see http://www.cgd.ucar.edu/csm/models/cp1/doc4 and http://www.cgd.ucar.edu/csm/models/cpl).

- It allows the model to be broken down into *separate components*—atmosphere, sea-ice, land, and ocean, that are "plugged into" the flux coupler ("drive"). Each component model is a separate code that is free to choose its own spatial resolution and time step. Individual components can be created, modified, or replaced without necessitating code changes in other components, unless new parameterizations require more information from the other components.
- It *controls the execution* and time evolution of the complete model by controlling the exchange of information among the various components. (It also provides rudimentary fault detection.)

It *computes interfacial fluxes* among the various component models (based on state variables) and distributes these fluxes to all component models while ensuring the conservation of fluxed quantities.

- It handles the *mapping* operations required to transform data between the different grids used in different components.

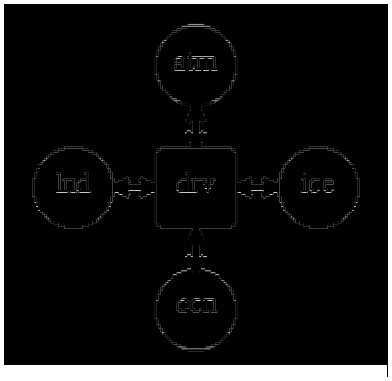

Figure 3. CSM Schematic

As illustrated in **Figure 3**, one way of thinking about the coupler is as a physically distinct process that arbitrates and implements all information flows among model components. This also turns out to be how the coupler is implemented in CSM-1 (but not PCM).

More abstractly, we can think of the coupler as a set of regridding, communication, and synchronization operations. This alternative view of coupling allows for more flexibility in implementation, as these various functions can be invoked wherever they make the most sense from a performance and software engineering viewpoint: within different model components or, alternatively, centralized in a distinct coupler process. The latter view is the one we adopted in the Avant Garde coupler toolkit design.

2.5.1 Coupling Design Issues

A number of issues complicate the design and implementation of coupling functions, in particular, the following:

- *Sequencing*. Corresponding timesteps of the coupler and the various component models may be executed in sequence or at the same time (concurrently). In the sequential approach, each component always has access to the latest state information from other components. The concurrent approach permits parallel execution, which is essential if components are to be distributed. However, it requires that one set of fluxes (typically the atmosphere) must be lagged by one timestep to achieve parallelism

between the component models. In addition, serial dependencies between components can result in idle processors.

- *Frequency of communication*. The various model components may exchange information at every time step or less frequently. There are obvious tradeoffs to be made between performance and accuracy.
- *Distribution*. The coupler and the various component models may be executed on the same processors ("stacked") or, alternatively, on disjoint sets of processors ("distributed"). The parallel approach can have performance advantages, as intercomponent communication tends to be less than intracomponent communication. On the other hand, the stacked approach avoids the need to compute efficient allocations of processors to components; if this is not done (or is not possible), then load imbalances occur. Note that hybrid approaches are possible: some components may be stacked (e.g., land and atmosphere in many models) while others are distributed.
- *Coupler parallelism*. Because different components use different grids and different representations of quantities of interest, the coupler can be required to perform considerable communication and computation. Hence, parallelism within the coupler can be important.
- *Separate or single executable.* The various model components can be linked into a single executable or (in a distributed approach) maintained as separate executables.
- *Support for standalone execution.* Individual model components will be used at different times in two different modes: as part of a coupled system or "standalone" with boundary conditions obtained from a file. A well-designed coupler can avoid the need for two different versions of each component, by allowing boundary conditions to be obtained via the same mechanisms in each case.

2.5.2 Coupler Performance Issues

The flux coupler controls the exchange of interfacial information between the component models of the coupled climate system. The primary and defining requirement of the flux coupler is that it must ensure that physical properties, such as momentum, energy, and fresh water are numerically conserved in the exchanges between component models. This requirement is complicated by the fact that different models use different types of grids at diverse resolutions. Thus the conservative regridding of interfacial fields between component grids is a key aspect of the flux coupler. As will be seen later, these regriddings present two serious software engineering problems. The first problem stems from the fact that the number of possible regriddings scales as the number of components squared. This "handshake" problem affects the complexity of the flux coupler in every respect. The second problem is derived from the fact that these regriddings are not especially well load balanced and not easily parallelized.

One of the factors limiting the performance of the current flux coupler is a load imbalance in the remapping of fields between component model grids. In a conservative remapping, grids cells on one grid must communicate with any grid cell on the other grid with which it overlaps. If both grids are simple latitude-longitude grids, then the communications pattern in the remapping is regular because both grids are regular. However, in the case of POP displaced-pole grids (see Figure 2), large communication imbalances occur for two reasons. First, POP grid cells can vary greatly in size; cells near the equator are much larger than cells near the poles of the grid. Second, the ocean grid cell that overlaps the North Pole covers all (128 for a T42 grid) the cells of the atmosphere grid that are converging at this pole point. Both of these result in cases where some grid cells need to communicate with a very large number of cells on the other grid while other grid cells need to communicate only with a few cells on the other grid. Such a disparity causes a large communication load imbalance.

In addition to regriddings, the flux coupler performs flux calculations. These calculations are typically performed on the finest grid in the coupled system for accuracy reasons. Thus a typical scenario for flux coupler operation is to receive a set of 2D state variables and input fluxes from a component on a particular grid. These fields are then regridded to the highest-resolution grid in the climate system, on which output fluxes are calculated. The resulting output fluxes are interpolated back to the original component grid and returned to that component.

Another complicating factor when designing efficient coupling schemes is the considerable dynamic range in the number of regridding calculations required, depending on problem resolution. For example, compare the relative costs of a $2/3^{o}$ ocean to T42 atmosphere regridding with a 3^{o} to T42 atmosphere regridding. The regridding cost varies according to the number of grid points, that is, by a factor of $O((3/(2/3))^2) = \sim 20$ in this case. Other combinations (e.g., high-resolution atmospheres and statistical models) will likely result in coupler performance demands that are dramatically different in other dimensions.

With respect to the mappings themselves, there are two issues: moving data (messaging passing latency and bandwidth issues) and load imbalances (sparse matrix multiply where work is not evenly distributed). Performance data indicate that message-passing costs exceed computational costs for a modest number of processor elements; furthermore, message-passing costs increase with the number of processor elements.

2.5.3 The PCM-1 and CSM-1 Couplers

The work at NCAR resulted in the development of two distinct coupling strategies. PCM-1 uses a sequential, stacked strategy, in which all model components execute in sequence on the same processors. CSM-1 uses a concurrent, distributed strategy, with ocean, atmosphere, ice, land surface, and coupler each executing simultaneously on disjoint processors. The CSM coupler uses the SCRIP remapping package from Los Alamos National Laboratory (http://www.acl.lanl.gov/lanlclimate/SCRIP), which implements the general remapping scheme of Jones (1999) for performing conservative remapping between any two grids on a sphere. In the case of surface fluxes, these remappings must be (and are) performed in a conservative manner.

Frequencies of intercomponent coupling, and numbers of fields exchanged are presented in **Table 1**. Coupling with the ocean currently occurs once per model day. Atmosphere, land, and ice couple much more frequently: between once per hour and once per atmosphere timestep (20 minutes). In general, land/atm/ice interactions with the coupler are at least 1 order of magnitude more frequent that in the case of the ocean. The following table summarizes the coupling frequencies and volumes to be used in CSM-2.

Table 1. Coupler component timing and flux/state exchanges.

Component	Coupling Frequency	Communication	
		Component -> Coupler	Coupler -> Component
Land Surface Model	Once per hour	6 states, 6 fluxes	7 states, 9 fluxes
Ice Model: CICE	Once every two hours	6 states, 13 fluxes	11 states, 10 fluxes
Atmosphere model: CCM	Once per hour	7 states, 10 fluxes	6 states, 6 fluxes
Ocean model: POP	Once per day	4 states, 3 fluxes	6 fluxes

Experiences with PCM and CSM illustrate some of the tradeoffs noted in the list of coupling issues above. In PCM, for example, we find that the model scales quite well up to 64 processors (on IBM SP and SGI Origin) but that scaling drops off beyond that point (**Figure 4**). This lack of scaling is a result of poor parallel efficiency within the more two-dimensional models used for the ice, land, and flux coupler, which in the PCM strategy must execute on all processors.

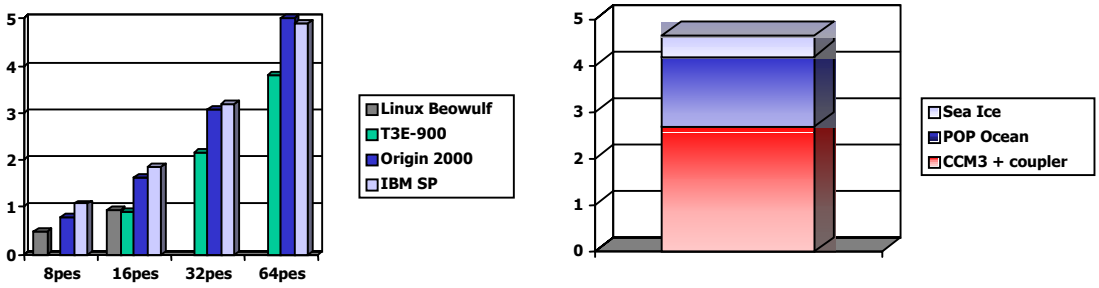

Figure 4: PCM performance in simulated years per wallclock day, on different computers (left) and wallclock hours per simulated year on a 64 PE SGI Origin 2000 (right) (T42 atmosphere, 2/3^{o} ocean).

With CSM-1, we encounter both load-balancing problems and scaling problems with the coupler. As an example of scaling problems, on 64 IBM Winterhawk-1 nodes, the atmosphere model at T42L18 takes around 30 seconds per model day (0.5 seconds per time step). In this configuration, the coupler running on a single processor takes 15 seconds per simulated day to execute the atmosphere critical path (the code that cannot be overlapped with atmosphere model execution) when coupling with a 3^{o} ocean. Hence, the

coupler needs to be sped up by a factor of 5 to get the overhead down to about 10%, and about 30 to get the overhead down below a single T42L18 CCM timestep. Multithreading has been used to a limited extent within the CSM-1 coupler but has not overcome this performance problem.

Thus, history left us two very different flux coupler designs, one threaded and one pure message passing. Neither implementation was particularly well suited to modern parallel systems composed of clusters of multiprocessors, which require a hybrid parallel programming approach for optimal performance. A significant design challenge for the Avant Garde project was to develop a flux coupler that is more scalable, more extensible, and less susceptible to changes in computer system design and capabilities.

3 Target Platforms, Model Configurations, Software Engineering

It is infeasible from a software maintenance point of view to have other than a single version of a model source code. Hence, an overriding goal of the Avant Garde project was to develop a *performance-portable* CSM: that is, a code that is able to execute efficiently on a variety of platforms and in a range of model configurations. It is nevertheless useful to identify initial model configurations and target platforms, with the goal of defining the space within which performance portability is required.

3.1 Target Model Configurations and Throughput Goals

The "standard" CSM configuration is currently 3^o (T42) atmosphere and 2^o ocean. DOE-sponsored simulations designed to provide input for regional climate studies are likely to require higher resolutions, at least 1.5^o atmosphere and $2/3^o$ ocean. NASA data assimilation and forecasting applications demand significantly higher resolution: $1/2^o$ atmosphere and $1/3^o$ ocean.

The target atmospheric resolutions appropriate to these applications range from 3 degrees in the horizontal for long-range climate studies, to 1/3 degree for process studies. The vertical resolution for the CSM-2 atmosphere component, CCM-4 (or CAM) will be 30 to 60 levels for the standard model and 100 levels when an active stratospheric model is incorporated. More aggressive resolution targets, such as suggested by the ACPI for support of regional climate studies, were not precluded from consideration, but were not feasible on the target machines. CCM-4 can carry a set of chemical species and tracers, some focused on physics and transport diagnostics (such as radon and 210-lead) and some to incorporate non-CO_2 gases and their influence on climate prediction (such as sulfate aerosols, methane, nitrous oxide, and ozone).

These numbers emphasize that a performance-portable CSM needs to be able to deal with a wide range of target resolutions, although it is not unreasonable to assume that when running on large numbers of processors the model will be run either at higher resolution or as part of an ensemble.

Our model throughput goals are geared to accelerate the development cycle. For the atmospheric model, the goals is achieving overnight turnaround on a 15-year atmospheric climate simulation evaluating the low frequency behavior and interannual response of the model. A similar productivity level for the ocean model is a 50-year simulation in 24 hours.

3.2 Target Platforms

DOE production computer platforms, as well as NCAR and other NSF center platforms, define our target architectures and also provided development cycles for this effort. The DOE centers at National Energy Research Scientific Computing Center (NERSC), Oak Ridge Center for Computational Sciences (CCS), LANL and LLNL have large high-performance systems of 64 to 512 compute nodes, with each node incorporating 1 to 64 processors in a shared memory subsystem. Within a node, shared-memory multithreading (e.g., OpenMP or pthreads) or within-node message passing are both possible.

For the Avant Garde project, CSM used three specific platforms: IBM SP, SGI Origin, and networked clusters (in particular, the Compaq AlphaServer at ORNL). But the ability to exercise the model on traditional vector supercomputers is also important to this effort. The design goal thus was to maintain performance portability across scalable parallel supercomputers, commodity clusters, shared-memory multiprocessors, and vector supercomputers.

Each of the four target architectures introduces distinct challenges for code development. A benchmark relevant to climate modeling resolutions was recently made by using a two-dimensional decomposition of the CCM3.6. This benchmark excludes output but incorporates full 3.6 physics with Eulerian spectral dynamics. **Figure 5** shows execution time per day on many of the targeted parallel systems.

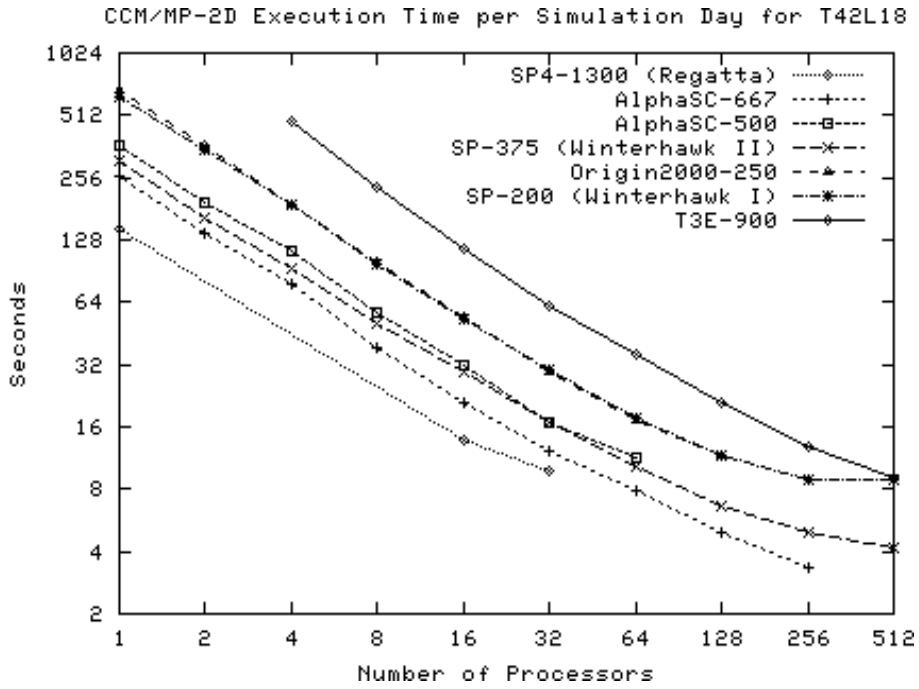

Figure 5. Computational platform comparison using a two dimensionally decomposed atmospheric model CCM/MP-2D.

Scalable parallel supercomputer. The most likely high-performance platforms for CSM-2 in the next few years are scalable parallel supercomputers such as the IBM SP. These systems feature O(1000) microprocessors connected via a high-speed switch, with these processors grouped in small-processor (4–64) shared-memory clusters. Though distributed-memory message passing using MPI is generally the most portable programming paradigm, it can inhibit code readability and maintainability. The use of a mixed distributed shared-memory programming paradigm introduces message passing only at the highest levels in the call tree and can be hidden from the scientists introducing new parameterizations. Optimization within a shared-memory node using OpenMP for parallelism and vector directives can provide the added parallel performance without affecting code readability. Excellent processor performance within a shared-memory node can be obtained if care is taken to manage cache utilization. In the past, the percentage of peak performance achieved on nonvector machines for climate, ocean and weather applications has been less than 10%. With 4 and 8 MB L2 caches becoming standard on scalable supercomputers, it may be possible to realize a higher percentage of peak processor performance, even on the very complex climate model calculations.

Commodity clusters. We can also expect to see CSM-2 used frequently, particularly in university settings, on small to medium-sized "commodity clusters" constructed by connecting high-end microprocessors (e.g., Intel Pentium or DEC Alpha) with fast networks such as Myrinet or Gigabit Ethernet. Performance issues here are similar to those encountered on scalable parallel supercomputers, except that potentially lower network performance and less sophisticated I/O systems may place additional demands on some model components.

Shared-memory multiprocessors. Another common platform for CSM-2 will be small to medium-sized shared memory multiprocessors (e.g., Sun, SGI). A performance-related issue here relates to how coupled models are configured: experience shows that a coupled model structured as a set of independent executables can perform poorly on such platforms because of high-context switching overheads.

Vector computers. While vector computers are not currently easily accessible to NCAR scientists, the performance and cost-performance of the current generation of vector supercomputers from Fujitsu and NEC remain impressive. Hence, it is important that CSM not abandon support for vector computers. Just

as the decision to code exclusively for vector caused problems when emphasis shifted to scalable systems, eliminating support for vector now will prove equally troublesome when barriers to acquiring the latest vector systems are overcome. Moreover, some CSM users currently have access to such systems, and future supercomputer architectures available in the United States may incorporate vector elements. In practice, this means that the model must be constructed to accommodate vector computations along a long, stride-1 data dimension. Our experience suggests, and we demonstrate in this project, that codes can be engineered flexibly to provide vector-friendly loop structure and storage order along with variable blocking factors for cache optimization (Michalakes et al. 1998; Michalakes 2000; Ashworth 1999).

Each platform also implements different input/output (I/O) systems. High-performance parallel I/O has been a weakness of many parallel machine designs. CSM generates large volumes of history and restart data during the course of a multidecade simulation. The I/O subsystems must also be taken into account and a general parallel I/O design, for history output and for restarts, incorporated in CSM.

4 Avant Garde Research and Development Results

We now describe the work that we performed in this project. As noted above, the primary goal of this work was to develop a performance-enhanced CCSM capable of meeting DOE simulation goals on DOE supercomputers. Hence, the bulk of the material in this section is concerned with the techniques used to enhance CCS2 performance at both the node and multiprocessor levels. Successful development of a performance-enhanced goal also required that the DOE/NCAR team accomplish a number of subsidiary goals relating to design documentation, performance portability, code readability, and community development infrastructure, and so these topics are also addressed.

Performance portability and code readability were critical because, to be successful, any changes made by DOE researchers to CCSM must be immediately folded into the core CCSM code base. If this was not done, then the "standard" CCSM and the "performance-enhanced" CCSM would quickly diverge. But in order for CCSM developers to accept DOE modifications, they must be designed to achieve performance portability—that is, good performance across a range of platforms—and must not significantly compomrise code readability and maintainability. As we will explain, we achieved performance portability via a mix of performance parameterization (e.g., block sizes in physics) and architecture-specific modules (e.g., spectral transform).

Design documentation and community development infrastructure are critical because one side effect of this project was to expand the community of CCSM developers significantly. There was a broad acceptance at the start of the project successful joint work requires a tighter coupling of NCAR and DOE (and NASA) developers than has been the case in the past. In order to achieve this tighter coupling, we developed detailed design documentation (so that modules, interfaces, and responsibilities are well defined) and established a shared code repository and standard testing mechanisms.

The principal tasks undertaken in the project are summarized in **Table 2**, taken from the original proposal but with the third column summarizing final status rather than indicating, as in the original, proposed completion date. As can be seen, the vast majority of proposed tasks were completed successfully. An unforeseen task was the parallelization of the code for sulfur cycle chemistry. This module is an important evolving science development in the CCSM. More details are provided in the rest of this report.

The scientific development of the code and the priority work of model developers did have an impact on this software engineering project. That impact came largely in the form of scheduling conflicts with the archive and diversion of key developers time. Coordinating the various tasks required in this project with the changing and unpredictable tasks encountered in the support of the scientific development was tricky. For example, the physics restructuring of the atmospheric code required modifying almost every loop in all the parameterization codes. It was feasible to schedule this sweeping change only when most scientific development was suspended. Also, when the Atmospheric Model Working Group tried to meet a deadline for simulations and modifications associated with the choice of dynamical cores and convection schemes, the repository updates for other purposes had to be suspended. This situation was expected (as was an adverse impact on the scientific development from our project). But the end result was a code that will realize better throughput and be more extensible will pay off for both aspects of the model development. In the meantime, the inconvenience encountered must be held partially responsible for the delays encountered in our project.

Table 2. **ACPI Avant Garde project task list with each task's status and a brief description. Items in boldface are additional tasks.**

Activity	**Description**	**Deliverable (Responsibility)**	**Final Status (Contributors)**
CCM preliminary design	A design of CCM defining goals and software interfaces	Document	Requirements (NCAR, ORNL)

		(NCAR)	Architecture(NCAR, ORNL), Parameterization Interface (NCAR)
CCM dycore interface	A modular CCM code that implements the designed interface	Preliminary modular CCM code (NCAR)	Designed and implemented (NCAR)
CCM-4 Utility and Math library	Library implementing data transposition, halo updates, and transforms for parallel decompositions	Tested code compatible with modular CCM (ORNL)	PILGRM library (LLNL,NASA), History module (NCAR,ORNL), Phys_grid and Dyn_grid (ORNL)
CCM Parallel Physics Design	Planning of the column physics package update to allow a specifiable run as the inner index, optimization for cache, optimization for vector.	Tested code and characterization of performance (NCAR)	CRM prototype experiment (ORNL)
Sulfur cycle parallelization and chunking	Unforseen task that needed to be carried with the continuing development of the archive.		Complete (ORNL)
CCM Parallel Physics Optimization	Implementation of the column physics package to allow a specifiable run as the inner index, optimization for cache, and optimization for vector.	Tested code and characterization of performance (ORNL)	Chunking from latitudes complete. General mapping incomplete. (ORNL,LLNL)
CCM (Lin-Rood) dynamical core	A scalable, parallel implementation of the Lin-Rood scheme conforming to the CCM-4 dycore interface	Tested, documented Lin-Rood code (LLNL)	2-D Block decomposition complete (NASA, LLNL)
CCM (Eulerian Spectral) dynamical core	A scalable, parallel implementation of the reduced grid Eulerian Spectral scheme conforming to the CCM-4 dycore interface	Tested, documented Eulerian/Spectral code (ORNL)	Blocked code incomplete(ORNL)
CCM (Semi-Lagrangian) dynamical core	A scalable, parallel implementation of the reduced grid Semi-Lagrangian/Spectral scheme conforming to the CCM-4 dycore interface	Tested, documented Lagrangian/Spectral code (ORNL)	Blocked code incomplete(ORNL)
CCM integration	A scalable, parallel implementation of CCM-4 which includes the three dynamical cores.	Scalable CCM tested and available (NCAR)	Ongoing (ALL)
Full CCM Software Engineering design	A full software engineering design exercise for the atmospheric component of CSM	Design document (NCAR)	Ongoing. Draft documents available. (See appendix for listing.)
POP (barotropic solver)	Performance improvement of the barotropic solver for coarse resolutions	Demonstrated performance in POP (LLNL)	WR direct solver examined in test mode, but no improvement realized. (LLNL)

Coupler performance study	Performance study of current coupler (PCM and CSM)	Performance report and documented ideas for improvement	Performance analysis completed, documented. (NCAR, ANL, LBL)
Coupler performance optimization	Performance optimizations to the coupler used within the current CSM	Performance demonstrated in CSM	Performance optimizations implemented, documented. (NCAR)
Preliminary Flux Coupler design	A preliminary design document to improve the scalability of the CSM flux coupler	Design document	Requirement, Architecture, Design documents completed. See Appendices. (NCAR, ANL, LBL)
CSM (coupler) Regridding Software	Parallel re-gridding software that effectively deals with load imbalances and scaling	Tested, documented coupler regridding code	Exists in MCT, tests in progress. (ANL)
CSM (coupler) Object Framework	An object oriented framework for the flux coupler that solves the handshake problem.	Tested, documented coupler framework code	Layered software design implemented. Tests in progress. (NCAR, ANL,LBL)
Coupler extensions	Investigations of coupling tropospheric chemistry with CSM	Design document	Incomplete.
Coupler software design	Software engineering design for CSM flux coupler		Requirements, Architecture, Design documents completed. See Appendix. (NCAR,ANL.LBL)
Source code repository	Develop centralized source code repository	Source code repository operational for CSM (NCAR)	CVS access and procedures implemented (NCAR)
Automated build/test infrastructure	Develop automated methods for building and testing CSM-2 components and entire model	Automated build-test operational (ANL)	Test-model.pl implemented (NCAR)
Model performance characterization	Detailed performance characterization of current CSM and PCM on NERSC computers	Detailed report (LBNL)	Timing utility library (NCAR). Studies (ORNL)
CSM I/O Framework	Incorporates parallel I/O support for history and restarts of component models	Implementation on target platforms and performance characterization (LBNL)	CCM History module (NCAR, ORNL), high performance I/O studies (LBNL)
CSM integration	Fully integrated scalable CSM which incorporates CCM-4, POP, coupler	Tested, documented scalable CSM code (NCAR)	CCSM-2 release delayed for science validation and improvements. Software incomplete.

5 High-Performance Atmosphere Model

During the 18 months of this project, the atmospheric model of the CCSM has undergone intensive development from both a scientific standpoint and a software engineering perspective. The CCSM Atmospheric Model Working Group conducted broad evaluation of new convective parameterizations and of three dynamical cores in an effort to define a standard configuration for CCSM2 simulations and code release. Cloud water was added as a prognostic variable. A major new long wave radiation treatment that includes up-to-date water absorption was added to the model as well as fundamental changes to the treatment of vertical diffusion of dry static energy. Sulfur cycle chemistry has been included as an option, and a completely new land model, the Common Land Model (CLM2) has replaced the LSM. The comparison and evaluation of the candidate improvements also required that a control configuration using the Eulerian dynamics with CCM3.6 physics be kept current.

Though these model developments were not part of this proposal, the software engineering and performance portability of the code required that the new and old modules be parallelized and their data structures reworked to conform to the design. The positive side of these unanticipated tasks is that performance benefits and code modularity will accrue to the model development. So for example, having the sulfur cycle code integrated with parallelism may encourage chemical studies. The negative impact of these extra tasks is that we did not complete all we had planned to do.

The effect of the additions of new models was minimized by the early concentration on the separation of dynamics and physics in the CCM3. The modularization allowed the finite volume work to progress nearly independently of the physics chunking work. This significance of swappable dynamical cores that this separation facilitates should be emphasized. To our knowledge, this is the first implementation in a global model of this concept.

The decision to pursue an object-oriented design and programming paradigm using the advanced features of the Fortran 90 standard was reaffirmed at many points of this project. The language support of modules is sufficient to our purposes, and developers quickly adapted to their use. The code has eliminated many of the legacy constructs associated with older versions of Fortran77 in the past 18 months in what would look like a complete rewrite from the outside. The conversion, however, is taking place without any downtime, time the model is not available for scientific development. The availability of reasonably good optimizing F90 compilers has aided us in pursuit of good performance.

Changes to the underlying data structures of the model required extensive modifications, and there are still further changes to come. Targeting cache, vector, and hybrid shared-distributed memory parallelism for performance portability may seem too broad a goal. But a good design will often run well on multiple platforms. We have not had the vector platforms to test on. But the model is regularly tested on the major DOE platforms in addition to the

Table 3. Column Radiation Model (CRM) performance study.

To examines issues of runtime loop bounds and compile-time array size declarations a test of the Column Radiation Model (CRM) was designed. Comparison is of the computational performance (in Mflops) for different compiler flags and values of the array dimensions. In the original compile-time experiments, -O3 produced consistently mediocre performance as PLON varied, while -O3, -qhot produced good performance when PLON > 16. Running on ORNL SP (375 MHz POWER3-II processors with 8MB L2 cache).

MFlops/sec PLON=1; PLOND=1,..,512; PLAT=512

PLOND	runtime		compile-time	
	-O3	-O3 -qhot	-O3	-O3 -qhot
1	111	75	115	112
2	113	74	114	112
4	110	73	111	109
8	102	70	103	103
16	85	62	85	89
32	71	58	72	79
64	47	46	45	62
128	22	29	22	35
256	11	16	11	17
512	6	9	6	9

So, CRM performance IS cache sensitive on the IBM SP3. If the columns being computed are widely separated in memory (PLON=1, PLOND >> 1) then performance becomes VERY poor. Compile-time and runtime performance is identical with -O3. With -O3 -qhot, compile-time is equivalent to -O3, while runtime -O3 -qhot performance is worse. (Not knowing loop bounds and poor cache locality leads -qhot optimizations astray?)

NCAR computers.

The software engineering processes that has been introduced to the CCSM model development is an extension of practices that the NCAR Core model development group used. An iterative test-design-implement cycle is best suited to the business of building scientifically valid coupled climate models. The practice with the atmospheric model has been extended and formalized to support multiple customers, DOE, NASA , NCAR and university researchers, as well as a distributed development team. In the first part of the cycle, we built a variety of prototypes to explore language and performance issues. This allowed the definition of standards on which designs could be based. At the end of this project, the fruit of the first cycle of implementations has begun to appear. It can be fairly said that the Avant Garde project has positioned the development to gather much greater returns than are indicated in this report. The project has pushed CAM development through one iteration of the software engineering processes.

5.1.1 Prototypes

Dynamical kernels such as FFT, spectral transform, and halo update were prototyped by using the hybrid programming paradigm and F90 modules in COWPOKE 1.0. It was concluded that inner loop OpenMP with outer loop MPI was efficient and that cost of "transposes" between calculation stages could be kept to the 5-10% overhead level.

Preliminary Lin-Rood and PCCM3 studies also indicated that the approach to parallelism was reasonable on target machine architectures.

Study of cache effects on Column Radiation Model (CRM) indicated that performance could be gained on cache machines by carefully choosing the data structure. Runtime vs. compile time parameters effects on performance suggest the dimensions of the structure should be specified at compile time, but the loop bounds could be set at runtime. (We would like to specify all at runtime for better portability, but the impact on performance was too great.)

5.1.2 Coding Standards

With the experience of the prototypes, a document *Coding Standard for CCM4* was developed as the coding style for all developers to follow. This provided the basis for the broader *CCSM Software Developers' Guide* (see appendix for the Web link).

5.1.3 Design Documents

The strategy followed for the development of the atmospheric component was to begin with a software engineering design for CCM-4. This design met near-term goals to incorporate scalable parallelism and modest encapsulation for a single source CCM-4. The design will be revisited and extended in a more complete "open design" process in future developments. We think this strategy offered the shortest path to single source PCM and CCSM, with the lowest adverse impact on CCM-4 development.

At the end of this project the process of adding new capability to the model has evolved towards a mature software management practice. Three classes of documents are being maintained to support the distributed team for the atmospheric model development. Initial public versions were made available.

First a *CCM4 Software Requirements* document outlines what the code needs to do. We have not written a scientific requirements document for the atmospheric code (though the coupler group did produce one), but restricted attention to the functional requirements of the code. This describes *what* the code must do.

Second, a *CCM4 Software Architecture* document that describes *how* the requirements will be met.

Third, a set of documents that offers *detailed interfaces and information on the implementation* of the architecture. Documents that were produced in this class include the physics parameterization interface, the dynamics and physics interface, calendar utility interface. Several other documents are planned in conjunction with ESMF project and continuing CCSM collaborative efforts. These are particular instances of the layered architectural design.

5.1.4 Testing and Repository Procedures

The testing procedures put in place for the atmospheric model have become the cornerstone of model development. This turned out to be one of the most significant contributions of the Avant Garde project and has greatly improved productivity of the development team. Erik Kluzek developed a set of PERL scripts that incorporated site-specific configuration details for ease-of-use. Using these scripts, one can build and test the latest version of the model at any of the participating centers by typing 'test-model.pl'.

The tested features expanded over the course of the project to include several model configurations: adiabatic idealized physics, full physics, the three dynamical cores. Multiple resolutions, restart, and different parallel decompositions are also exercised in the tests.

For the majority of developer changes, the test script reports that output is 'bit for bit' compared with output of the previous version. This is a validation and quality check that the developer has not introduced a bug and that scientists can expect exactly the same climate to be produced from the code after the change.

Some changes affect the numerical algorithms used in the code and thus may affect results at the level of the round-off error of the finite-precision machine. To validate this class of changes, the testing script compares results of a perturbation test and reports whether the error growth is within the acceptable and predicted rates.

No attempt was made to automate the scientific validation of the model as a result of incorporation of new or improved physical parameterizations. This task lies outside the scope of a software engineering project.

The use of the testing procedure on all repository changes allowed us to quickly identify problems and correct them. Hence, the working CCM code has remained working and available for scientific development through the course of the project.

Atmospheric Test-Model Script

The test-script runs the following tests for each dynamics (Eulerian, Semi-Lagrangian, and Lin-Rood). Eulerian and SLD are run at a fairly low resolution of T21 with 18 levels. Lin-Rood is run at a 4x5 degree resolution with 18 levels. In both cases the resolution is deemed high enough to do a reasonable test, but low enough to give a faster turnaround.

All.) Test with trace-gas.

All tests are run with "trace_gas" set to true so that code handling multiple constituents will be checked.

0.) First run a few time-steps with the DEBUG flag set to true.

This runs with the debug option on and turns bounds checking on and other features needed for debugging. This test is also run in two parts. It's run with and without SPMD-mode turned on, and the answers are compared to ensure that SPMD and non-SPMD both give identical answers.

1.) Restarts are bit-for-bit (and change decomposition).

a.) Test restarts

Restarts are done at an odd time so that both history and abs/ems restart datasets will be produced. The restart test is also run with a different number of nodes than the initial run to ensure that restarts don't depend on the parallel domain decomposition.

b.) Changing the domain decomposition.

This is done by subtracting one from the previous number of shared memory CPU's used, and dividing the number of SPMD nodes by two.

2.) Compare restart comparison to previous version.

At the end of the test all of the control tests are examined to see if the results are bit-for-bit identical to those computed using the previous library. If so the script mentions this fact, if not, a WARNING message is printed as well. If answers are different, the user must determine if the differences are acceptable.

3.) Error growth test.

Running with a perturbation of zero and round-off level (1.e-14).

The results of this test need to be plotted after the test is done by using the "ccm/bld/graphgrowth.csh" utility

4.) Compare to previous version.

Finally, if the "-c" command-line argument is given the results for the error growth test with a perturbation of zero is compared to the results for a perturbation of zero with the previous source code (given after the "-c" option). Results can be compared to the error growth profile produced to determine if they are roundoff. If "-c" is NOT given, but it was previously and the files are still available, it still does the comparison.

5.) Adiabatic mode on and off.

The last two tests (3 and 3) are run both with adiabatic mode ON and OFF. This ensures that the model works in both modes.

5.1.5 Physics-Dynamics Separation

To promote the modular development of the atmospheric model and to demonstrate swappable dynamical cores (for the first time), a great deal of effort has gone into separation of dynamics and physics in the

model. This exercise enabled a unique experiment in the comparison of dynamics models by the CCSM Atmospheric Model Working Group (AMWG). The comparison of the Eulerian-spectral, Semi-Lagrangian-spectral and finite volume dynamical cores supports the decision process of what to include in the next release of the model. Though the comparison work was not part of this project, the software design and code work allowed for comparison between dynamical cores using the same physics package. These comparisons were performed in the fairest setting possible.

The separation required that an interface with two types of dynamics algorithms be accommodated: the time split and the process split update algorithms. The two spectral packages use a processes split algorithm, while the finite volume code uses a time split method. This high-level design will accommodate all the dynamical cores currently under development.

An object-oriented design for the physics and dynamics separation was partially implemented. In particular, each dynamics package is testable with an idealized physics option so that development and source code is strictly separate from the physical parameterization code. The data structures of the dynamics object were at one time pervasive through the entire model, so that changing or altering the dynamics structures required physics modifications as well. These are now clearly separated and, in fact, different for the dynamical cores. Parallel decompositions may be different between the dynamics and physics.

A grid transpose (or regridding) step is required between dynamics and physics packages to correctly couple the parallel, distributed data of the two packages. The coupling interface was encapsulated in a module, DP_COUPLING, which will allow for further optimization and machine-specific tuning without affecting other parts of the code. What has been implemented at the end of this project cannot yet take advantage of an arbitrary plug-and-play dynamics package. The mechanisms transposing between data structures have not included a general N to M shuffle as conceived in the design. Certain assumptions have been enforced that require that the dynamics processors be a subset of the physics processors. These assumptions will be relaxed or removed in the future.

5.1.6 Physics Chunking

A chunk is an arbitrary collection of atmospheric model columns. The modification to the physical parameterization packages making chunks the underlying data structure was extensive. But the change affects only the computational performance of the model. Results of the model were unchanged to the level of bit for bit reproducibility from our implementation of chunked physics.

This new data structure was introduced as a generalization of the two-dimensional parallel decomposition available in the PCCM3 and the one-dimensional parallel decomposition available in the CCM3. It facilitates two performance gains: increased scalability to utilize more processors and better cache sizing to increase memory bandwidth. The chunked data structure allows the use of a reduced grid or full Gaussian grid in the spectral dynamical cores.

The performance improvements that can be realized because of this change have not been fully explored. This is particularly true in the context of the new computer architectures with more complex memory hierarchies, such as the new IBM Power4. What has been done is a study on the Compaq AlphaServer and the IBM SP Power3, currently the highest performing platforms we have available for testing.

Figures 6 and 7 plot the tuning results when using four OpenMP threads on the IBM Winterhawk II and the Compaq AlphaServer SC. Both full model times and time spent in physics routines are plotted as a function of the chunk size NCOLS. The time spent in physics was measured by using the CAM thread-safe timing library with the TIMING_BARRIERS option enabled, which inserts barriers between phases of the programs. The full model timings were collected with the TIMING_BARRIERS options disabled. In all cases, the model runtime did not increase significantly when running with TIMING_BARRIERS enabled.

We ran the tuning experiments with 16 MPI processes and either 1 or 4 OpenMP threads per process. The Compaq and IBM systems are both clusters of 4-way SMP nodes. 16 nodes were used for both the 1 and 4 OpenMP thread experiments. Thus, in the 1 OpenMP thread experiments 3 processors per node were idle. 64 processors (16 nodes) were chosen for the tuning experiments as this is the largest processor count that can be used and still have at least one chunk per processor when using the "traditional" 128-column chunk.

The 4 OpenMP thread experiment utilized OpenMP parallelism over the "local" chunks, ranging from 4 to 128 for NCOLS=128 and NCOLS=1, respectively.

In **Figures 8 and 9** the throughput performance as a function of chunk size for the atmospheric model is compared on both platforms. The Semi-Lagrangian and Eulerian performance are reported. The highest throughput is achieved using the Semi-Lagrangian version with a chunk size of eight on the Compaq AlphaServer.

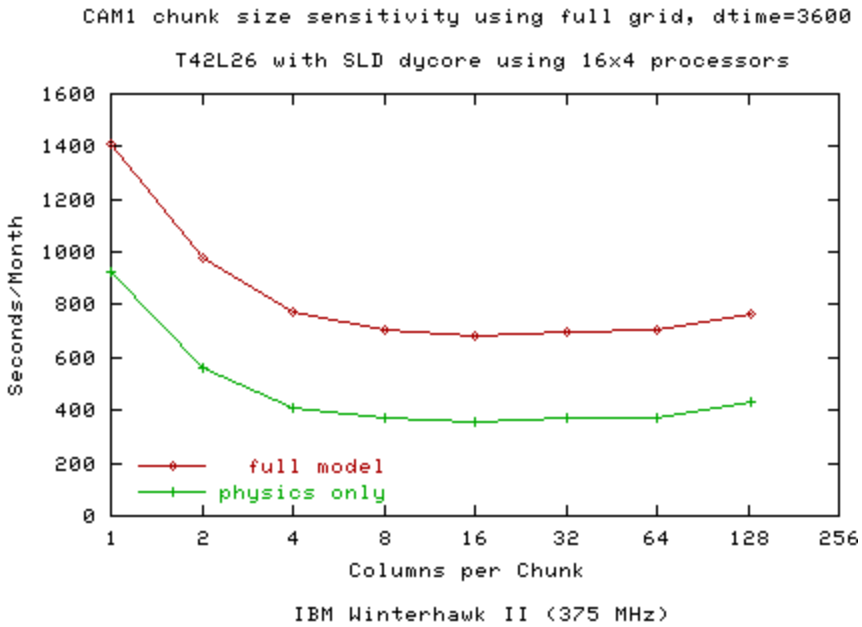

Figure 6. Full model and physics throughput for the CAM on the IBM SP3 (WHII)

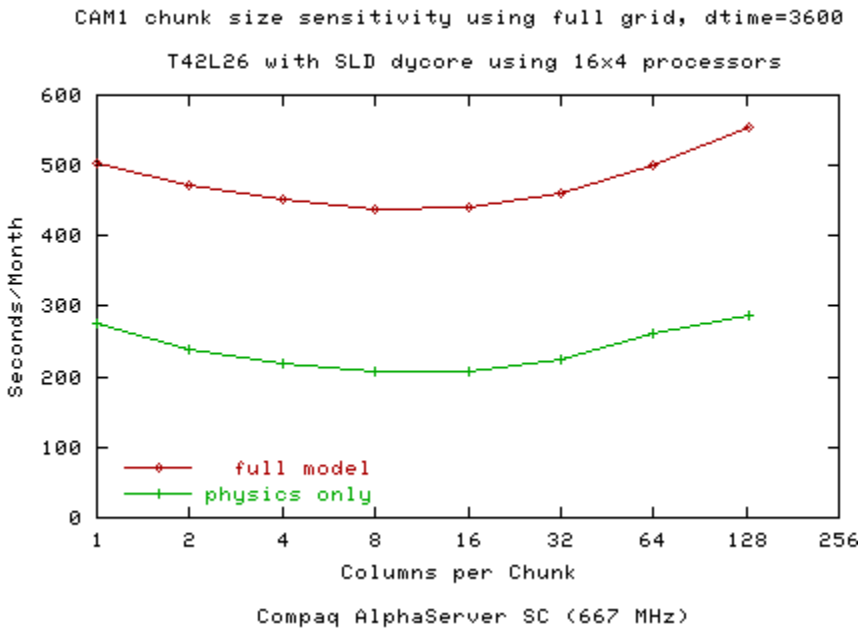

Figure 7. Full model and physics throughput for the CAM on the Compaq AlphaServer

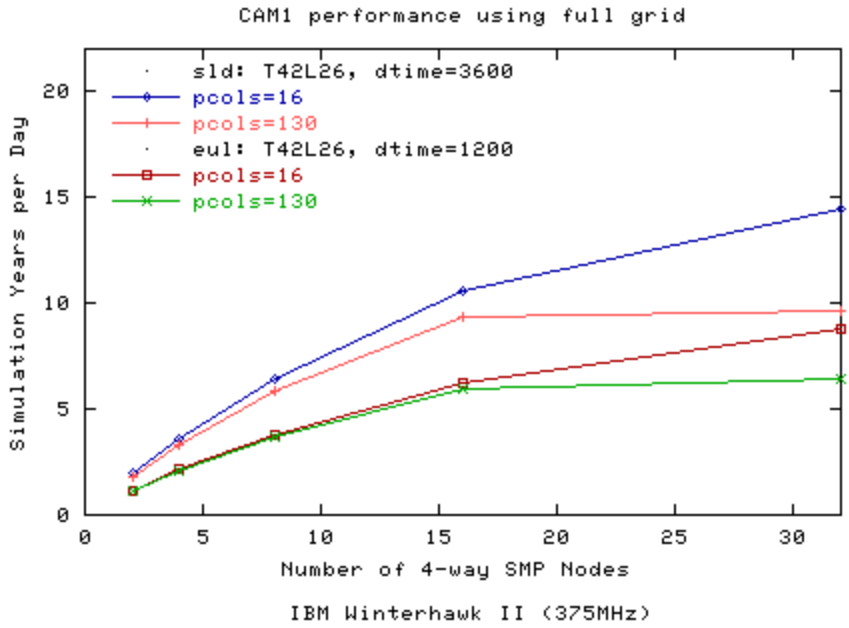

Figure 8. CAM Semi-Lagrangian and Eulerian throughput, varying chunking parameter on the IBM SP3 (WHII).

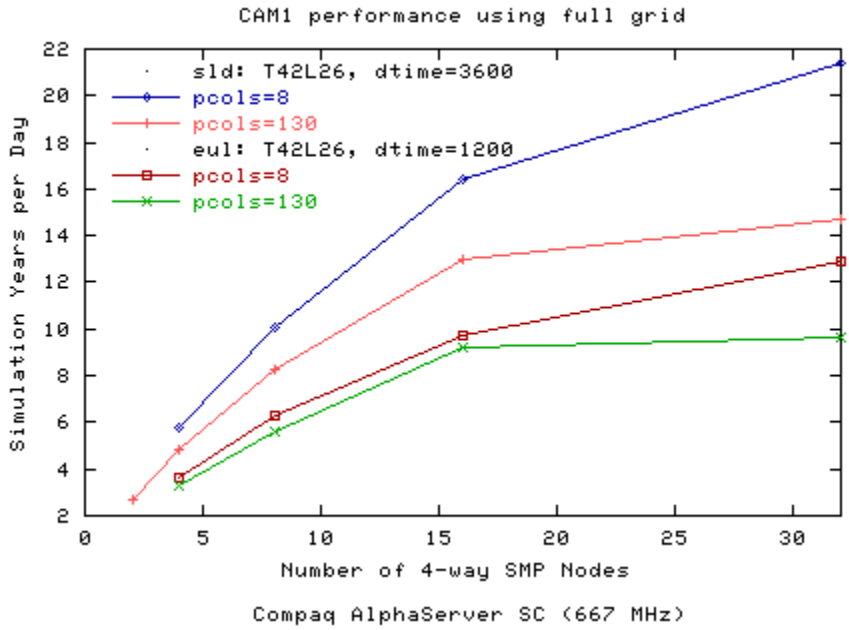

Figure 9. CAM Semi-Lagrangian and Eulerian throughput, varying chunking parameter on the Compaq AlphaServer

A complete set of plots and performance study is available on the Web at the following URL: (http://www.csm.ornl.gov/evaluation/CAM).

5.1.7 Dynamics Blocking

After the physics chunking, the dynamics blocking introduces increased scalability and parallel performance for the rest of the atmospheric code. According to our design, a block is a geographically contiguous set of model grid points. In the case of the spectral cores, this block also includes extensions (or

halo regions) to allow semi-Lagrangian departure point calculations and interpolations on processor. The three dynamical cores are each working with some level of introduction of blocking in their data structures. The finite volume dynamical core is furthest along, having achieved a two-dimensional parallel decomposition into blocks.

Scalable Parallelization of Lin-Rood Dynamical Core

A scalable parallelization of the Lin-Rood dynamical core has been under development, in two phases: (1) implementation of multiple distributed memory two-dimensional domain decompositions, and (2) restructuring of shared memory parallelization to take account of the higher dimensionality of the decompositions. The first—and largest—phase is essentially complete.

The nature of the semi-Lagrangian control volume technique leads to a vertical decoupling of much of the dynamics. This lends itself to a domain decomposition in both latitudinal and vertical coordinates. The Lagrangian surface remapping algorithm, however, strongly couples the vertical dimension while maintaining columnar independence, thereby lending itself to a domain decomposition in both longitude and latitude. Our approach thus uses a latitude/vertical decomposition for the main dynamics and a longitude/latitude decomposition for the remapping, with the two decompositions connected by using high-speed transposes provided by NASA's PILGRIM library. The PILGRIM library has been added to the CCM utility layer and demonstrates the extensible, layered software architecture that we have adopted.

This approach was successfully implemented and tested in NASA's finite volume code, where for the dynamics alone a three-fold increase in throughput was attained over what could be accomplished with a decomposition in latitude only (**Table 4**). The algorithm has since been installed in CCM and is undergoing testing.

Additional parallelization is attained through shared-memory constructs. The fact that most of that parallelization is in the vertical direction, which coincides with one of the coordinate directions of the distributed-memory parallelization, limits the degree of shared-memory parallelism attainable. The second phase focuses on improving the shared-memory parallelization. One possibility is to parallelize (using shared memory) jointly in latitude and level. Another possibility is to parallelize in longitude. Use of longitude offers greater opportunity for parallelism; however, effective cache utilization is more challenging, as longitude is generally the fastest turning array index.

Table 4. Lin-Rood performance study for two dimensionally decomposed dynamical core.

Times are for dynamial core only (excluding physics) on NERSC IBM GSeaborg machine for 2-deg x 2.5-deg x 55-level test case, for different latitude/vertical domain decompositions. The number of vertical subdomains is shown horizontally, and the number of latitudinal subdomains is shown vertically. Limiting to a latitudinal-only domain decomposition restricts use to 18 computational nodes. A two-dimensional domain decomposition allows use of at least 108 nodes and decreases the computation time by a factor of 3.

Lat / Vertical	1	2	4	6
1	995			
2	512	329	175	
6	189	127	76	
9	140	93	53	
18	89	60	38	31

An additional focus is on the computation of the geopotential, which tightly couples the vertical levels. The geopotential is calculated during the subcycled portion of the dynamics and necessarily involves nonlocal communication. It is therefore critical that the geopotential calculation be carried out as efficiently as possible.

Introduction of Blocks in the Eulerian and Semi-Lagrangian Spectral Dynamical Cores

An additional index has been introduced in the physical grid data structures of the two spectral dynamical cores. This changes the memory-processing organization from a latitudinal slice to a block. A block decomposition for parallelism then replaces the one-dimensional latitudinal decomposition. The design of the block interface to the dynamical cores is discussed in the atmospheric design documents.

Implementation is currently limited to checking that the introduction of the additional index does not break the code. Exercising parallelism and introducing a two-dimensional decomposition comprise the next step but will apply only to the computations in physical space. Further parallelization of the spectral space computations will be addressed in the follow-on SciDAC project.

5.1.8 Atmosphere-Land Surface Interface

During the course of this project, the LSM model was replaced by the CLM2 land model. The domain decomposition of the land model is entirely independent of the atmosphere, even when both are hybrid MPI/OpenMP parallel with internal coupling. The price for this general coupling is that data is gathered prior to transmission in each direction. Dealing with these interface and coupling issues still needs to be done in the context of the new coupling technologies and performance improvements in the atmospheric model. We need to eliminate the requirement for a gather before communication with the coupler, for all components and regardless of the coupling method. And chunking data structures need to be an option for use of the land model.

6 High-Performance Flux Coupler

A major goal of the Avant Garde project was to design and develop a next-generation coupler. Specifically, the design of such a coupler was to address the two major challenges identified in the origin proposal: performance portability and software extensibility.

The coupler is the model component responsible for coordinating the execution of the various components that form the coupled system. Its four primary functions are readily identifiable: (1) allow the fully coupled model to be broken down into separate components, which consist of a variable number of diverse codes contributed by individual groups, (2) control the execution and time evolution of the coupled system, (3) provide the mechanisms responsible for exchanging information (e.g., interfacial fluxes that assure energy conservation) between the components, and (4) handle the mapping operations required to transform data between the different grids used in different components.

The predecessors to the next-generation coupler were very different in their fundamental designs. One (CSM1) was strictly a shared-memory, threaded code, while the other (PCM) was a distributed, pure message-passing code. Neither implementation was particularly well suited to modern parallel systems composed of clusters of multiprocessors requiring a hybrid parallel programming approach for optimal performance. In fact, the first phase of this project included an analysis of the current coupler designs and performance characteristics. A by-product of this effort was to identify possible performance bottlenecks and to propose, test, and implement changes in the current couplers that made an immediate noticeable difference in the performance of each. This work is summarized in the next subsection.

The second phase of this project was the development of a design of a next-generation coupler that would have the following properties:

- Support for sequential and distributed execution, and perhaps for hybrid configurations in which some components executed concurrently while others executed sequentially.
- High-performance scalable parallel implementation of performance-sensitive regridding and communication operations, enabling CCSM to execute efficiently on a large number of processors.
- Flexible configuration enabling run-time rather than compile-time specification of processor allocations.
- Support for standalone execution of individual component models with the coupler used to facilitate boundary forcing from independent sources (e.g., input files representing observations).

These properties formed the foundation in developing the requirements and architectural design of the next-generation coupler.

As was proposed, work on the next generation-coupler followed a coordinated software development process. This effort included the identification and review of the next-generation coupler requirements (computational and scientific), specific design documents based upon the criteria listed in the requirements document, and the creation of a central repository for unit test codes and prototype coupler improvements. To facilitate open and frequent communication, a centralized Web site was constructed to track task assignments and progress, frequent telephone conferences between participants were arranged, and an e-mail discussion forum was initiated.

6.1 Phase I: Performance Studies and Optimizations

One important goal of this project was prompt performance improvements to both PCM and CSM. This work was performed early in the project, while we were in the requirements and design phases for the next-generation coupler. These performance enhancements were not only of value to PCM and CSM users; they served as prototypes for components of the Model Coupling Toolkit.

6.1.1 PCM Coupler Enhancements

As outlined in the original proposal, one initial task involved evaluating the PCM coupler in order to elucidate its performance characteristics and bottlenecks. This benchmarking was performed by NCAR team members, and complete results of this performance study are available on-line at the URL http://www.cgd.ucar.edu/ccr/bettge/ibmperf/timers.txt. The PCM benchmark comprises the operations performed by the PCM coupler during a ten-day simulation using PCM. The operations included are: 1) interpolation of fields from the atmosphere grid to the ocean grid (2160 calls in 10 days), and 2) interpolation of fields from the ocean grid to the atmosphere grid (1440 calls in 10 days), and global and hemispheric integrals on both ocean and atmosphere grids used to enforce energy and water conservation.

Many of the classic, well-known coupling problems were identified. It was determined that regridding performance was limited more by data movement (message passing) than by the sparse matrix multiplication load imbalance and that the entire regridding function scaled very poorly. The flux conservation technique used by PCM1, involving global reductions, scaled poorly and consumed a significant amount of the coupler's time. Flux calculations are essentially one-dimensional physics computations and scale well but can be poorly load balanced when grid masking is involved.

Team members from LBL focused on the three most time-consuming functions in the PCM coupler – flux conservation, ocean-to-atmosphere regridding, and atmosphere-to-ocean regridding. By combining global vector-vector dot-product together, communication overhead was saved. A detailed analysis of the send/receive communication patterns in the regridding functions resulted in lower latency and improved performance. A 30-40% speedup in each of the individual subroutines was achieved. The overall coupler timing was improved by 7% on 16 processors, 10% on 32 processors, and 20% on 64 processors. These results are documented on the NGC Web site.

Team members from ANL also studied the communications pattern associated directly with the matrix-vector multiplication used to implement interpolation between component model coordinate grids. PCM coupler performance figures (http://www.cgd.ucar.edu/ccr/bettge/ibmperf/timers.txt) showed that communications costs associated with matrix-vector multiplication were well in excess of any load imbalances, so optimization efforts centered on the communications pattern.

The total time to send a message containing S bytes is

$$T = L + S / B.$$

Here T is the total time (in seconds) to send the message, L is the latency (in seconds), and B is the machine bandwidth (in bytes/second). The latency and bandwidth costs for a message are equal for a message of size $S=BL$, which we call the *critical size*. For typical message sizes significantly smaller than BL, message latency is the dominant cost, and the strategy for speeding communications is, where possible, to combine multiple messages into a single message to reduce latency costs. For typical message sizes significantly larger than (or even the same order of magnitude of) BL, the bandwidth costs dominate (are appreciable), and a message compression strategy, if applicable, is appropriate.

Bailey (2000) cites bandwidth and latency values for numerous parallel platforms, and from these figures it is possible to compute values of BL **(Table 5).**

Table 5. Communication parameters for computational platforms. Numbers cited for the SP-3 were measured by C.H.Q. Ding of NERSC.

Platform	Latency L (μs)	Bandwidth B (MB/s)	Critical Size BL (bytes)
IBM SP-1	50	25	1250
IBM SP-2	35	40	1400
Cray T3D (PVM)	21	26.9	565
Sun E6000	11	160	1760
Cray T3E (MPI)	17	300	5100
IBM SP-3 (colony switch)	25	470	11750

The message-passing strategy for the PCM matrix-vector multiplication **y** = **Mx** is as follows:

1. The vectors **x** and **y** each have their own decompositions.
2. Determine which vector is longer. If **y** is longer than **x**, decompose **M** by row, and then gather the elements of **x** (based on the decomposition defined in step 1) necessary to compute the elements of **y** owned by a given process.
3. If **x** is longer than **y**, decompose the **M** by column, compute the partial sums for elements in **y** owned by a given process, and then send these partial sums to the appropriate processes in the decomposition of **y** from step 1.
4. In steps 2 and 3, the messages and communications schedule is as follows: If process B needs *any* data from process A, send *all* the data owned by process A to process B. This approach has the advantage of being easy-to-code and relatively safe, but has the disadvantage of communicating excess data. All communications operations are blocking.

In the PCM coupler, multiple fields are interpolated in many of the calls. Often, six fields were interpolated, which implied that the messages sent during the interpolation process were in excess of the critical size *BL* defined earlier in this section. For example, interpolation of six fields between a T42 (8192 points) to a POPx1 grid (110512 points) on an SP-3 (8-byte real number representation) results in message sizes of 25476 bytes on 16 processors, 12288 bytes on 32 processors, and 6144 bytes on 64 processors. Clearly a message-passing strategy should yield improvements.

The excess data communicated in the fourth step implies an opportunity for performance optimization. If this excess data is significantly larger than—or even a significant fraction of—the critical size *BL* defined above, then a message-packing strategy is in order. Our analysis began with a precise measurement of the *minimum* amount of data each process required from each other process to perform the matrix-vector multiplication. In many cases we found one to two orders of magnitude less data needed to be communicated, and that the typical excess data amounts were significantly larger than the critical size *BL*. These measurements were made an integral part of the PCM coupler code and then used to construct smaller, packed messages.

The impact of this strategy was positive, as can be seen by Jumpshot[1] timeline visualizations of a single-field atmosphere T42 (8192 points) to ocean POP (110592 points) interpolation before (**Figure 10**) and after (**Figure 11**) these changes were implemented. Note that in Figure 8 the excess data communication created *communications* load imbalances that far exceeded any computational load imbalances. The message-packing strategy not only reduces the typical communications time per processor but also substantially reduces communications load imbalances. Similar performance improvements resulted for the ocean-to-atmosphere interpolation scheme.

[1] Jumpshot is an MPI performance analysis and visualization tool developed in the MCS Division of Argonne National Laboratory. Information is available on-line at the performance visualization web site http://www-unix.mcs.anl.gov/perfvis/software/viewers/index.htm .

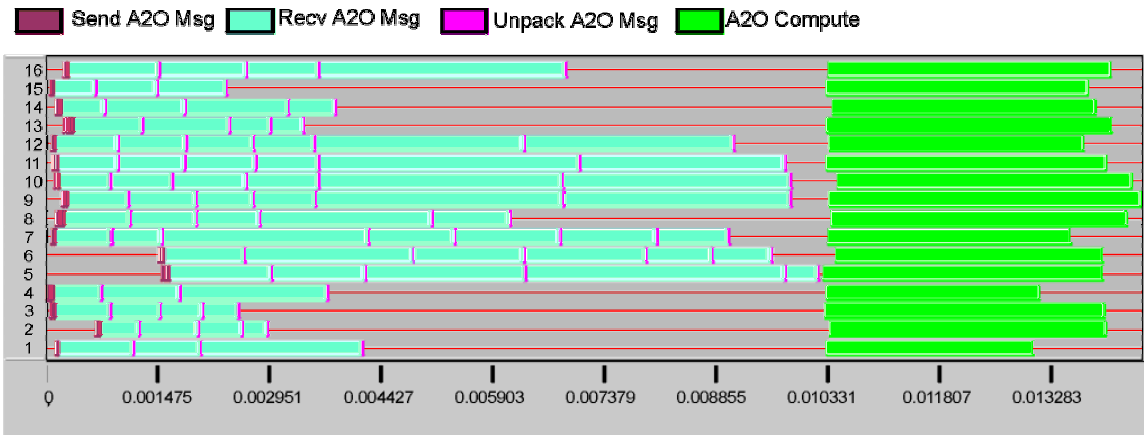

Figure 10. Jumpshot plot for the matrix-vector multiplication for the atmosphere-to-ocean interpolation of a single T42 field to a POPx1 field in PCMon 16 processors (original communications algorithm).

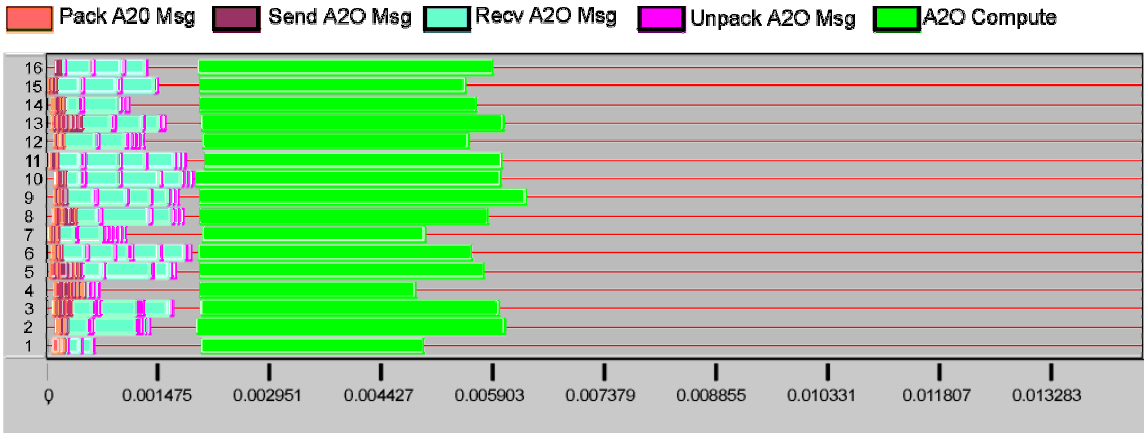

Figure 11. Jumpshot plot for the matrix-vector multiplication for the atmosphere-to-ocean interpolation of a single T42 field to a POPx1 field in PCM on 16 processors (using packed messages).

Once the bandwidth-related messaging costs were minimized, effort was focused on hiding some of the communications time through the use of nonblocking communications. The use of nonblocking communications yielded some performance gains, as can be seen in **Figures 12 and 13**, which show the impact of nonblocking versus blocking communications in the matrix-vector multiply for the ocean-to-atmosphere interpolation in PCM. This strategy yielded a speedup by approximately one-third over the blocking communications using packed messages—a figure in agreement with the LBL results for implementing nonblocking communications in PCM without message packing.

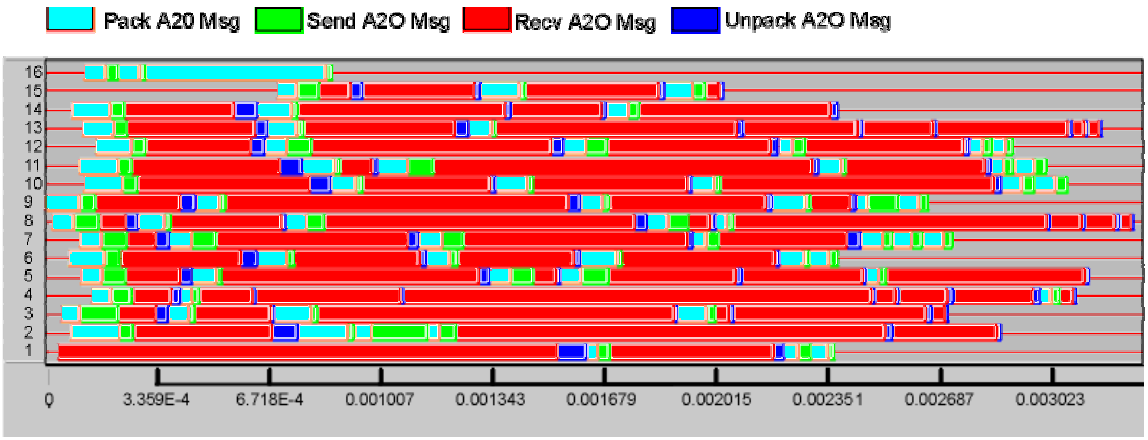

Figure 12. Jumpshot plot for the matrix-vector multiplication for the ocean to atmosphere interpolation of a single POPx1 field to a T42 field in PCM on 16 processors using packed messages and blocking communications.

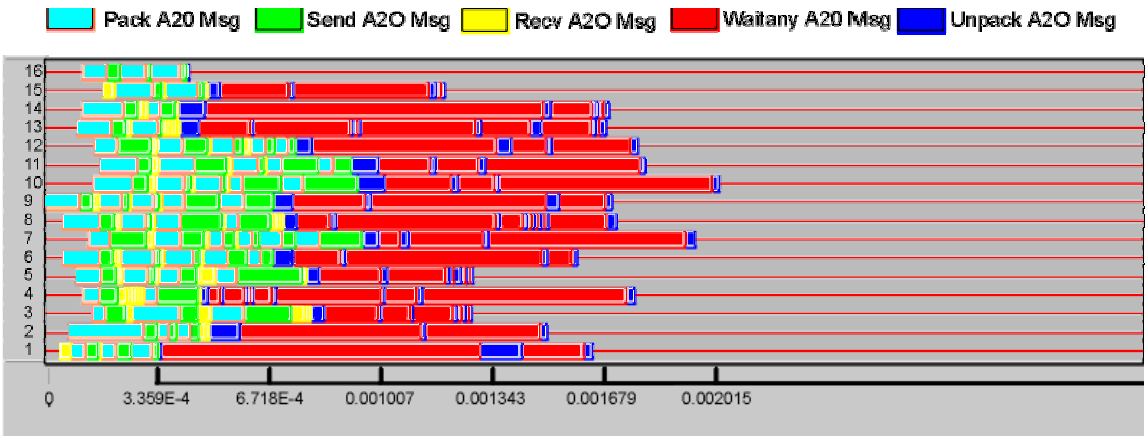

Figure 13. Jumpshot plot for the matrix-vector multiplication for the ocean-to-atmosphere interpolation of a single POPx1 field to a T42 field in PCM on 16 processors using packed messages and nonblocking communications.

Overall, the implementation of packed messages with nonblocking communications in PCM yielded significant performance improvements for process pool sizes typically used for production simulations**.** In **Table 6,** we present the speedup on various process pool sizes for the atmosphere-to-ocean, ocean-to-atmosphere, and total coupler time. These figures are compiled for the interpolation between the atmosphere T42 grid and ocean POPx1 grid. The speedup factor reduces with the increase in the number of processors. This is no surprise because for a fixed problem size, the number of points in a subdomain decreases as the number of processors is increased. This results in a decreased subdomain area-to-perimeter ratio, and a corresponding decrease in the amount of excess data communicated in the original PCM regridding algorithm. This does not necessarily make the coupler a bottleneck, because for process pool sizes in excess of 64, the atmosphere and ocean models also experience diminishing scaling returns for the T42 and POPx1 grids, respectively. The crucial factor in judging the performance gains for the PCM interpolation scheme is the amount of excess data communicated in the original PCM algorithm. Just as increasing atmosphere and ocean resolutions will allow these models to scale better, so will the improved version of the PCM coupler.

Table 6. PCM Coupler timing profiles measured before and after introduction of message-packing and non-blocking communications. Measurements are made for a ten-day PCM coupler benchmark. All times are given in seconds.

Number of Processors	Atmosphere-to-Ocean Comms (original)	Atmosphere-to-Ocean Comms (optimized)	Ocean –to-Atmosphere Comms (original)	Ocean-to-Atmosphere Comms (optimized)	Total Coupler Time for One Model Day (original)	Total Coupler Time for One Model Day (optimized)
16	13.364	2.919	14.542	2.059	129.163	112.038
32	15.807	2.393	17.138	1.441	134.379	106.553
64	11.653	2.188	15.090	1.234	138.789	118.653

6.1.2 CSM1 Coupler Enhancements

In a fashion similar to the analysis of the PCM coupler, the CSM1 coupler was subjected to a performance critique during the first phase of the project. The CSM coupler presented a different challenge because the code is based upon a shared-memory, threaded paradigm and is executed concurrently with the other models composing the coupled system. It became a high priority to examine the CSM1 coupler when it was announced that CSM1 would be using a distributed-memory, clustered SMP architecture for work due in June 2001.

The NCAR team members first determined that the CSM1 coupler's shared-memory restriction would not be a bottleneck to the entire coupled system performance. Nevertheless, the main sequencing loop was restructured to improve load balancing, and several communication bandwidth and latency issues were discovered and addressed. The most important changes involved coupler/ice exchanges, where temporal averaging and grid masking reduced the volume of data being moved by a significant factor.

6.2 Phase II: Design and Implementation of the Next-Generation Coupler

A major mission of this project was the design and development of a new CCSM flux coupler using modern software engineering practices. In this section we describe the development process for the new CCSM coupler.

6.2.1 Next-Generation Coupler Requirements

The first step in the development of the next-generation coupler was to prepare a list of requirements that state explicitly what the software must do. Such a document was prepared and reviewed by a select panel of coupled model experts.

The coupler requirements take into account the overall goal of the project: to achieve and maintain performance portability across a range of platforms in a variety of configurations. They state explicitly that the coupler should maintain the functionality of its predecessors, the PCM and CSM1 couplers, and extend their capability to meet the overall goals. The requirements are separated into two categories, scientific and computational functionality requirements, and contain a list of impositions that may be created inherently on the components that comprise the coupled model. These specific requirements formed the basis for the design of the next generation coupler.

The requirements can be found in the Next-Generation Coupler Requirements Document, available on-line at http://www.cgd.ucar.edu/csm/models/cpl/cpl-ng/. The documents have been subject to a review process that includes comments from other software engineers and scientists involved in the model development.

6.2.2 Overview of Design Considerations

An analysis of the scientific and computational requirements of the coupler yielded several broad conclusions:

- The coupler has both high-level command/control functions and low-level functions. There are distinct parts of the coupler that must know the details of the component models, while other parts need little knowledge of individual components (e.g., matrix-vector multiplication). This suggested a layered design strategy.
- The ability to support coupling for both sequential (PCM) and concurrent (CSM) model execution suggested a need to implement the coupler as one or more libraries that support the construction of either type of coupler.
- All component models as well as the coupler require coordination in setting up and sharing information about intermodel communication. This suggested that a generalized initial communicator group tool or library be built.
- Since there was no immediate need to support regridding of data on three-dimensional grids, the initial version of the coupler implements regridding of data on two-dimensional grids. However, the software is not restricted to two-dimensional grids.
- The demand that the coupler and its utility routines provide a Fortran90 API suggested that it was easiest to implement the coupler application layer in Fortran90.
- The requirement for flexibility in the fields passed to the coupler, as well as the extensibility to incorporate new component models, suggested a need for the coupler to have an internal data representation such as a Fortran90 derived type.
- The requirement that the coupler be capable of receiving and delivering data on numerous grids, including unstructured grids, suggested that the coupler have its own general scheme for describing gridded data.
- Since the first version of the coupler requires that all component models communicate via the coupler, the communications between component models and the coupler had to be implemented in parallel. That is, a general M-to-N parallel transfer was needed rather than the current gather-send-scatter approach that had been used in PCM and CSM1.

Based upon these conclusions, it was clear that the best design strategy was to assimilate and/or build a set of tools or libraries that can be used to create a wide variety of coupler applications.

6.2.3 Survey of Preexisting Geophysical Coupler Software

As part of a rational software development process, we asked ourselves the following question: "Can we modify pre-existing software to build the new CCSM coupler?" During the early stages of the design process, we studied many different public-domain software packages used for coupling geophysical models. A survey of these candidates is available on-line at the Next Generation Coupler web site at the URL http://www.cgd.ucar.edu/csm/models/cpl-ng/coupler_reviews . Software packages considered included: the Ocean-Atmosphere-Sea Ice-Surface (OASIS) coupler developed by CERFACS, the Goddard Earth Modeling System (GEMS), Climate Ocean Weather Parallel Object Kernals (COWPOKE), Parallel Library for Grid Manipulations (PILGRIM), and the Flexible Modeling System (FMS) developed by GFDL. We discovered much of this is high-quality software, but there were various obstacles to using these codes in the new coupler. For example, the OASIS coupler did not support distributed-memory parallelism. GEMS looked promising, but we were concerned about the learning curve involved in adopting it and using it for our application. COWPOKE provided some of the functionality we needed, but we were not convinced it had a sufficiently general M to N parallel transfer facility to suit our purposes. PILGRIM contained much of what we needed for the parallel transfers and regridding facilities, but we were unsure at the the time of the survey (August, 2000) that it was being supported. FMS looked promising, but its source code was not available at the time of the survey, and it was unclear that FMS would be supported for non-GFDL users. We also examined a piece of commercial software—the Mesh-based Parallel Code Coupling Interface (http://www.mpcci.org). MPCCI offered much of what we needed for the new coupler, but two things gave us pause: 1) no support for a flux/state data merging facility, and

2) the code is distributed *only* as a library—that is, the source code (written in C++) is not available (which we feel contradicts directly the spirit of a *community* model).

We also considered the large body of software frameworks developed by the Department of Energy. In particular, we were interested in the mature Parallel Extensible Toolkit for Scientific Computation (PETSC), as well as the under-development work of the Common Component Architecture (CCA) project and the Parallel Application Work Space (PAWS) project. We concluded that PETSC has useful functionality to offer for the regridding functions of the coupler, but we decided to defer this application until we could formulate a modular design of the coupler. In future work, we may use PETSC to address this issue. We concluded PAWS offered some promising features for the problem of parallel data transfer, but did not consider it sufficiently mature for our use. We feel the CCA offer many interesting features, and look forward to trying some of them in the coupler once the CCA project (now funded under SciDAC) has a mature release suitable for our purposes.

Our final conclusion was, given the time constraints of this project, that one or more new software tools were needed to develop the Next-Generation Coupler.

6.2.4 Overall Architecture of the Next Generation Coupler

The overall coupler architecture is a layered design. **Figure 14** summarizes the coupler software layers.

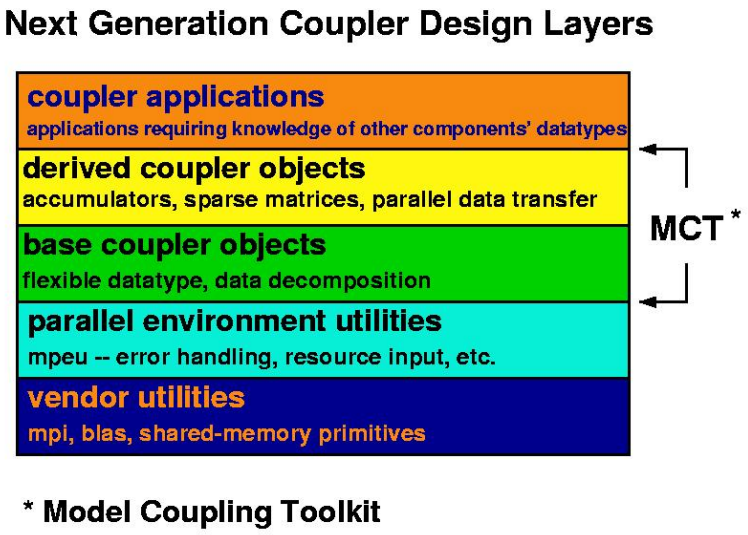

Figure 14: The next-generation coupler design layers.

The layers, ranked lowest-level to highest level, are the following:

- **Vendor Utilities**—standard or vendor-supplied utilities. They include such software as the Message Passing Interface (MPI) or the Basic Linear Algebra Subroutines (BLAS).
- **Parallel Environment Utilities**— utilities that provide access to MPI and support for error handling, diagnostic output, and runtime input of resources for the coupler. This layer also supplies some of the classes used to build the basic coupler data classes. Currently, this layer is served by the NASA Data Assimilation Office's Message Passing Environment Utilities (MPEU) library.
- **Basic Coupler Classes and Methods**—the internal data representation and data decomposition for the coupler.
- **Derived Coupler Classes and Methods**—the tools necessary for the coupler functionality that is hidden from the component models. This layer includes the parallel communication and parallel regridding infrastructure (MCT), as well as the communication handshaking tool (MPH).

- **Coupler Applications**—the top layer where a variety of couplers can be built using lower-layer utilities. Applications in this layer have information regarding the data types and data decompositions used in the component models being coupled. It also includes utilities for messages sent/received to/from the component models. It also contains high-level coupler application functionality (event-looping, history file management, etc.) that could not be abstracted to a lower level.

6.2.5 MPH: Multiprogram-Components Handshaking Utility

To facilitate the communications between the component models and the coupler, it is necessary to coordinate in a general fashion the way in which component name registration and the initialization of communications are handled. We developed a library for flexible communications between independent components in a distributed multicomponent environment. MPH is the multicomponent handshaking library that resolves these tasks in a convenient and consistent way.

MPH contains the functionality to handle the following situations, as required by the next-generation coupler:

- Multiple executables, each executable containing a component model, with no mix of components into any single executable image. CSM1 uses this execution mode.
- Single executable, all components stacked into a single executable load image, with component models sharing the same communicator. PCM uses this execution mode.
- Multiple executables, with at least one executable image containing multiple component models. Component models retain individual name and domain, and there is coupled overlap on processor subdomains. CCSM2 is designed to offer this mode as an option.

Source codes, a comprehensive description, and user guide are available at the MPH Web site.

6.2.6 MCT: Model Coupling Toolkit

A key to the development of the next-generation coupler was recognition that the coupler performs the function of data flow between component models and those basic functions could be abstracted and embodied in a software toolkit capable of supporting many instantiations of coupler applications at the highest level. Thus, the concept and implementation of the Model Coupling Toolkit (MCT) became a reality.

The MCT is a set of Fortran90 modules and routines that provide core services common to coupled models:

- Domain decomposition descriptors and global-to-local indexing. The toolkit supports both a simple 1-D decomposition and a general segmented decomposition capable of supporting multidimensional grid and unstructured grid decompositions. The toolkit provides means for initializing and querying these descriptors, along with support for exchange of these descriptors between component models.
- A registry for component models that describes the processes on which they reside. This allows for automatic translation between local (component model) and global process ID numbers needed for intercomponent model communications.
- A datatype for use as a communications scheduler, which can be initialized automatically from a pair of domain decomposition descriptors. This datatype allows for the automation of the complex parallel intercomponent model data transfers necessary for high performance.
- A proprietary flexible, extensible, and indexible data field storage datatype. This allows the user maximum flexibility in configuring a coupled model. If the component models and coupler are designed properly, the fields passed between component models could be configured at runtime. It is possible to store both real and integer data in this datatype, which allows a user to carry around associated gridpoint indexing information as a means of monitoring processing and an aid to debugging coupled model applications. The storage order in the derived type is field-major, which is compatible with the notion that most communication and computation in a coupler is processing of multiple fields of *point data.* This allows, where possible, for the field storage

datatype to be used directly as a buffer for MPI operations and allows for reuse of matrix elements during the interpolation process, thus boosting performance for this process. Facilities exist to support sorting of data, which can also improve single-processor performance through better cache usage during the interpolation process.

- Flux and state data field interpolation via an efficient sparse matrix-vector multiply that works directly with the MCT's proprietary field datatype. The MCT has a datatype for storing sparse matrices, along with their global and local row-and-column indexing information. Facilities in the MCT exist for sorting of matrix elements, which can yield a more regular memory-access pattern and thus boost single-processor performance through better cache usage.
- A datatype for describing coordinate grids that can support multidimensional and unstructured grids. This datatype allows the user to store geometric information such as grid-cell cross-sectional area and volume weights, and integer data for any number of grid indexing schemes.
- A flexible, extensible, and indexable datatype that serves as registers for temporal averaging and accumulation of state and flux data
- Routines for merging flux and state data from multiple component models for use by another component model, with fractional area weighting and/or masking encapsulated either in the coordinate grid datatype or the field storage datatype
- Support for computation of global integrals needed to ensure energy and moisture conservation in inter-model flux transfers

All of the datatypes in the toolkit have, as needed, transparent support for both point-to-point and collective communications operations. This allows the user to transfer complex F90 data types in a single communications call analogous to MPI. The support for collective gather/scatter/broadcast operations allows the user to checkpoint the coupler and component model interfaces with ease.

The MCT is build upon the NASA DAO Message Passing Environment Utility (MPEU), which provides support for basic low-level data types upon which MCT classes are built, Fortran90 module-style access to MPI, and tools for error handling, sorting, timing, and load balancing. Both MPEU and MCT have their documentation built in, implemented as extractible prologues compatible with the software package ProTeX, which translates the prologues into LaTeX source code.

The primary design of MCT is described in the MCT Design Document, and the programming interface is contained in the MCT API Definition Document. These documents are available via the MCT Web site http://www.mcs.anl.gov/acpi/mct .

Currently, the MCT is in beta release. A complete formal release of the MCT will be accomplished late in 2001.

MCT PerformanceStudies and Application in the PCM Coupler

The toolkit has been tested successfully in unit testers supporting both the asynchronous and event-loop paradigms for coupled models. We have demonstrated good scalability for the intercomponent model parallel transfer for suitably chosen domain decompositions on the source and destination models. The flexibility of the toolkit allows the user to choose any domain decomposition for a grid on each component model, and the choices of decompositions affect performance. We present two examples, one an "ideal" case in which the two decompositions are identical (**Figure 15)**, and one in which the domain decompositions are very different (**Figure 16)**.

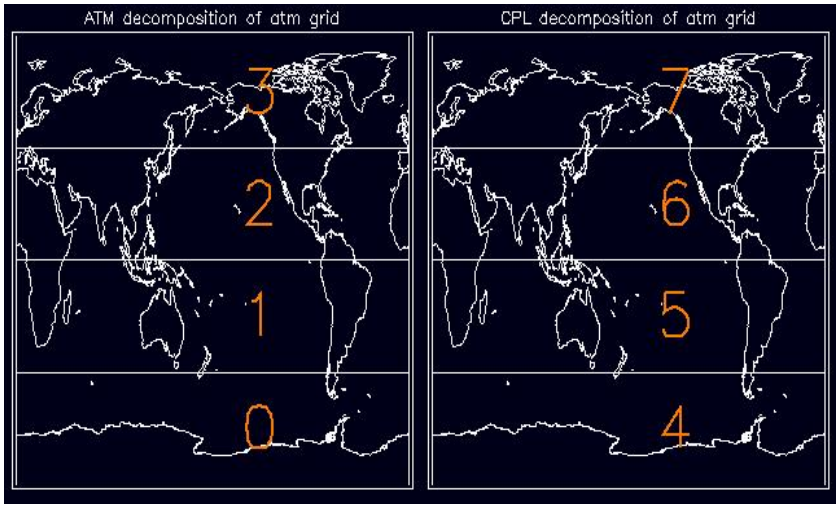

Figure 15. Domain decompositions of atmospheric field data for the atmosphere component model (left panel) and coupler (right panel). Both decompositions are identical, one-dimensional latitudinal decompositions.

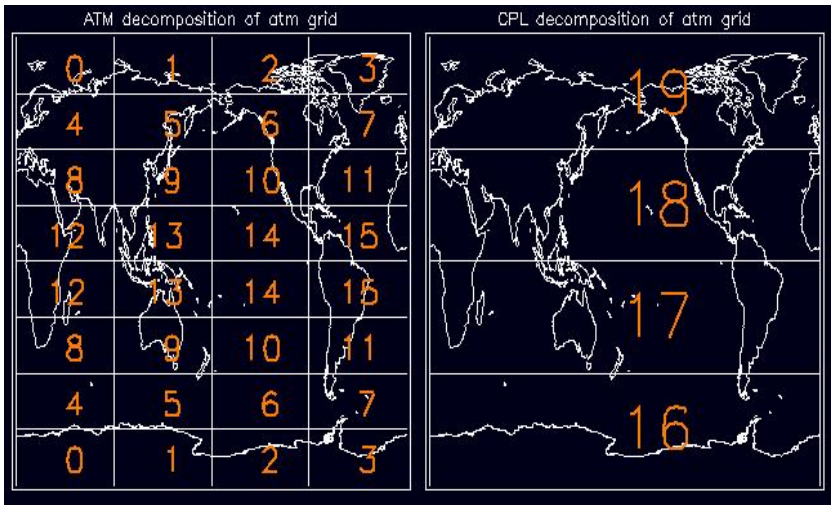

Figure 16. Domain decompositions of atmospheric field data for the atmosphere component model (left panel) and coupler (right panel). In this case the atmosphere is decomposed on sixteen processes using a two-dimensional, folded-equator decomposition. On the coupler the decomposition is a one-dimensional latitudinal decomposition.

The performance for the parallel data transfer for these two examples is quite different. In the identical decompositions case shown in **Figure 15**, the speedup with increasing numbers of atmosphere and coupler processors is more or less linear, as can be seen in **Figure 17**. The solid curve corresponds to the **`MCT_Send()`** operation. The dashed curve corresponds to the **`MCT_Recv()`** operation, which takes slightly longer because this operation must wait until all the data have been copied into the output field storage datatype. Scaling studies for the differing decompositions shown in **Figure 16** are presented in **Figure 18**. In this case, the atmosphere has a very different decomposition than the coupler and the number of nodes assigned to each is not the same. The **`Router`** between these two decompositions was automatically determined by MCT. The number of coupler nodes was varied for each of three cases: with the atmosphere on 8 (white), 16 (red), and 32 (blue) nodes. The poor scaling may be an unavoidable result

of doing a parallel data transfer between two very dissimilar decompositions. But the overall transfer time is still very small compared with the time the full model will spend computing 10 time steps, and the MCT user (the coupler developer) is relieved of determining the complex transfer pattern.

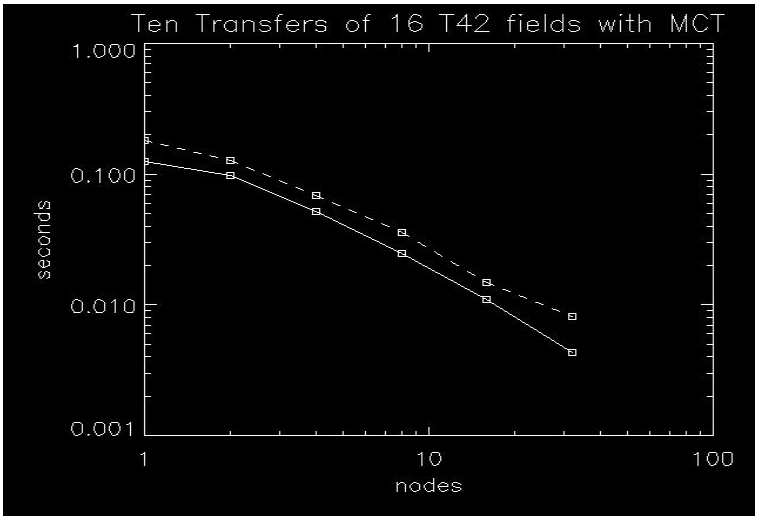

Figure 17. Scalability of parallel data transfer of sixteen T42 fields between atmosphere and coupler with domain decompositions shown in **Figure 15**. Data are shown for each component model having 1, 2, 4, 8, 16, and 32 processes. The solid curve shows timings for the MCT parallel send routine **`MCT_Send()`**, the dashed curve timings for **`MCT_Recv()`**.

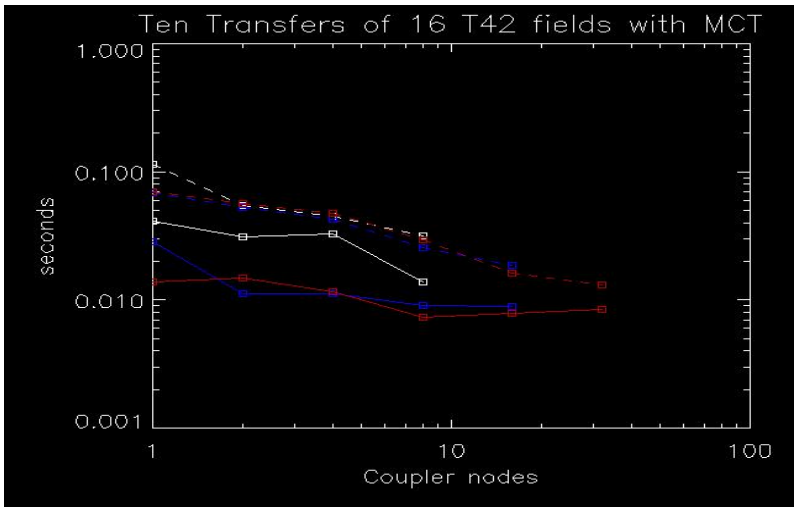

Figure 18. Scalability of parallel data transfer of sixteen T42 fields between atmosphere and coupler with domain decompositions shown in **Figure 16**.

We performed a performance study using the PCM coupler benchmark, but using MCT components. Timing results for the 2160 atmosphere-to-ocean and 1440 ocean-to-atmosphere regridding calls are presented in **Table 7**. The communications routing mechanisms in the MCT are far more flexible than those in the hand-tuned PCM coupler, but the atmosphere-to-ocean and ocean-to-atmosphere communications costs are either the same or slightly lower. The computation costs appear to be no worse or even better than the original PCM timings. This is quite encouraging considering the hand-tuned PCM used f77-style, static arrays, and the MCT implementation is using derived types built on top of allocated arrays. Both the PCM and MCT implementations have been designed so that the loop-order in the interpolation is cache-friendly The most likely reasons for better computational performance in the MCT

implementation are (1) the use of *local* rather than *global* arrays for the computations, which results in more compact memory references, and better cache-line re-use, and (2) the application of sorting of matrix elements in the MCT implementation, which also increases cache performance.

Jumpshot plots for individual calls to the atmosphere-to-ocean and ocean-to-atmosphere interpolations are presented in **Figures 19 and 20**, respectively. Each depicts a single call to the respective regridding function for six fields, transforming data between a T42 atmosphere grid and a POPx1 ocean grid.

Table 7. Coupler timing profiles using MCT components. Timings are measured for 2160 T42 atmosphere to POPx1 ocean calls and 1440 POPx1 ocean to T42 atmosphere calls. Timings are measured in seconds.

Number of Processors	Atmosphere-to-Ocean Communications	Atmosphere-to-Ocean Computations	Ocean-to-Atmosphere Communications	Ocean-to-Atmosphere Computations
16	2.909	4.408	1.809	1.616
32	1.609	2.644	1.359	1.180
64	1.452	1.936	1.156	0.984

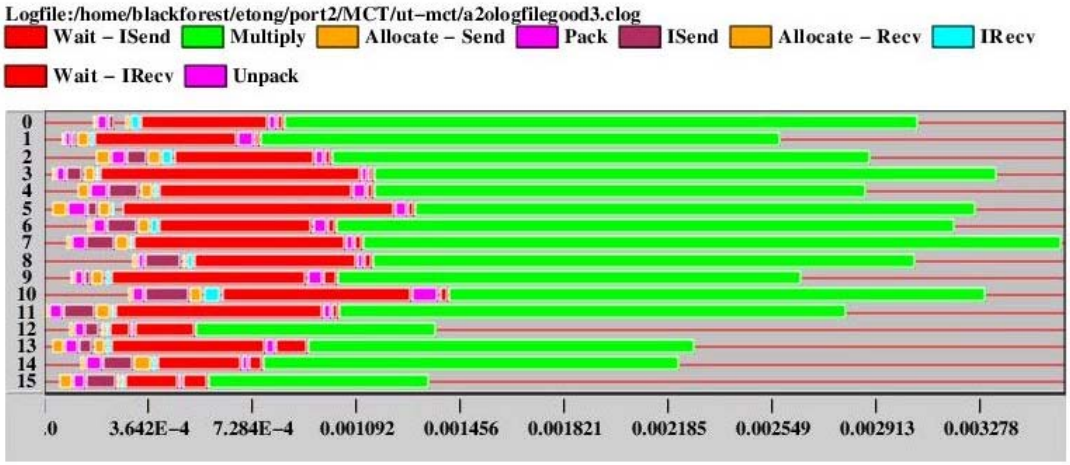

Figure 19. Jumpshot plot for the matrix-vector multiplication for the atmosphere-to-ocean interpolation of a single T42 field to a POPx1 field in MCT implementation of the PCM coupler on 16 processors.

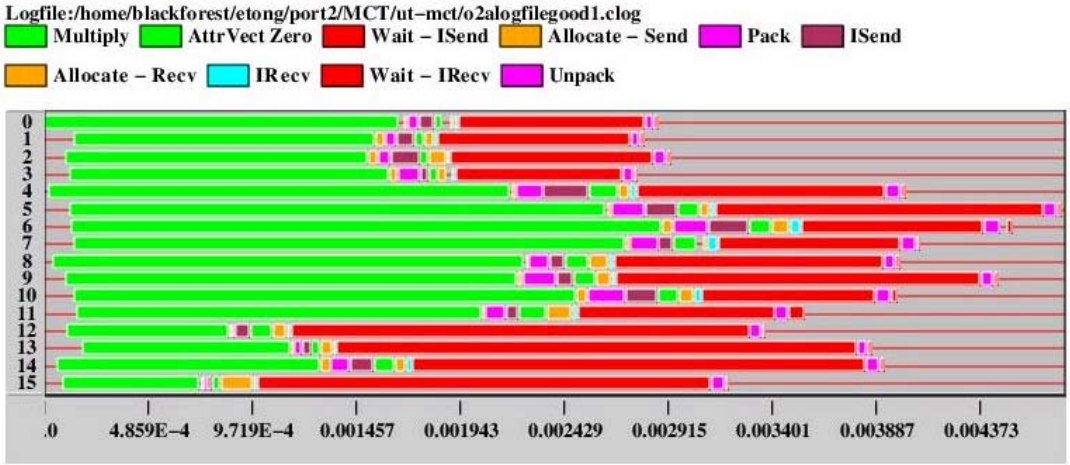

Figure 20. Jumpshot plot for the matrix-vector multiplication for the ocean-to-atmosphere interpolation of a single POPx1 field to a T42 field in the MCT implementation of the PCM coupler on 16 processors.

These performance results are promising, but show room for further improvement. One enhancement that will be made in the next version of the MCT is the inclusion of hybrid parallelism through the use of OpenMP directives in the multiplication step of the regridding. Another more immediate problem to address is that of load balance. Now that we have succeeded in streamlining the communications in the interpolation operation, computational load-balance during matrix-vector multiplication is now the dominant performance problem. This problem will be studied during the successor to this project, the SciDAC CCSM project.

6.2.7 CCSM CPL6

Work has begun on the instantiation of the next-generation coupler, CPL6, that will serve as the successor to the CCSM2 coupler (CPL5). CPL6 is the top-level coupler application built on top of the infrastructure provided by MPH and MCT.

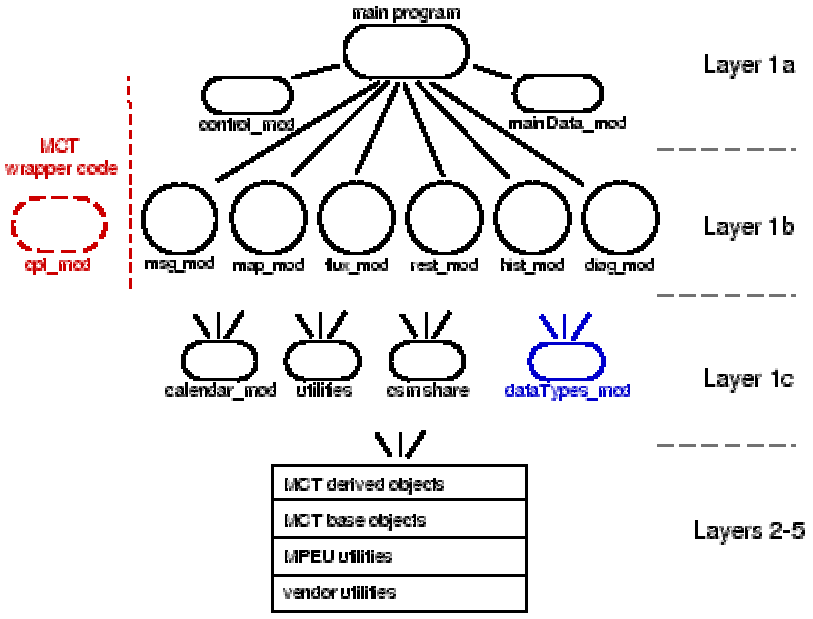

Figure 21. High-level design of the CPL6 coupler application.

CPL6 introduces a sublayering structure that focuses on the details that lie above the MCT and MPH level. **Figure 21** illustrates this view, grouping MCT, MPH, MPEU, *et cetera*, together with vendor-supplied libraries into the lowest layer. Layers 1a, 1b, and 1c are central to the design of CPL6, along with the MCT wrapper code, which is provided to component model developers to include as their interface to the CPL6 coupler. Consistent with the overall concept of the next-generation coupler philosophy, the component models will use MCT and MPH when communicating with the coupler, but their use will be hidden within the MCT wrapper code.

The CPL6 sublayers contain the functionality that is most closely tied to the command and control duties of the coupler application – event looping (or sequencing) of the component models, I/O of restart or history datasets, diagnostics, calendars, coupled system share software (e.g., solar zenith angle computation). By necessity, the CPL6 application layer also contains a module to define its own classes, using the derived data types defined in MCT.

CPL6 will follow the computational configuration of its predecessors in CCSM. That is, it will execute using the multiple program multiple data (MPMD) paradigm. It was recognized, however, that use of the MPMD configuration on the class of machines that are configured as distributed memory clusters of SMPs creates new, perhaps unique, load balancing issues that can be addressed only by the user (as opposed to being solved by the operating system on shared-memory vector parallel machines, for example). To

address these issues, the coupler team enlisted the assistance of Dan Anderson of the Scientific Computing Division at NCAR. He developed a set of unit testers that simulate the possible configurations of MPMD applications on the CCSM2 target machines.

CPL6 is currently under development by the coupler team at NCAR. The design considerations of CPL6 are summarized in the CCSM Coupler Architecture Version 6 Document.

6.3 Current Status and Future Schedule

As a result of the work accomplished under this project, the underlying foundation for general coupler applications has been established, namely, the MPH and MCT utilities. They resolve model coupling issues that were identified and categorized early in the project.

The coupler application for CCSM2 (CPL6) currently exists in prototype form. It is expected that the first release of CCSM2 (end of calendar year 2001) will contain CPL5. CPL6 will continue undergoing development under the funded SciDAC proposal *Collaborative Design and Development of the Community Climate System Model for Terascale Computers*. Under the proposed work, the MPH and MCT utilities will be further refined as the coupler team at NCAR develops the beta version of CPL6 and provides feedback to the team members responsible for the utilities. It is expected that CPL6 will be installed, tested, and validated in CCSM2 by early in calendar year 2002.

7 Improvements to POP Barotropic Solver Performance

The preferred method for solving the barotropic equations in the Parallel Ocean Code (POP) code, as used at LLNL, appears to be the preconditioned conjugate gradient (PCG) method applied to corresponding system of linear equations. This method does not, in general, exhibit desirable parallel performance characteristics. Also, although the POP code typically uses only two different coupling matrices throughout a run, the PCG method does not allow effective reuse of information as solutions are found for different source terms. Accordingly, an investigation has been started concerning other solver strategies that may have better parallel scaling and performance.

A new method has been proposed for solving linear systems arising from partial differential equations (i.e., having local stencils.) This method uses domain decomposition and the local stencil pattern to eliminate variables from the interior of each domain, yielding a reduced system of equations that can be solved with reasonable parallel scaling. Additionally, since in the case of POP many of the elimination operations need be performed only once, much overhead can be amortized over the course of many updates. In the work reported here, the reduced system is solved exactly, meaning that the method is a direct solve rather than iterative. The new method uses wave front recursion (WFR) to form the reduced system. This is the multidimensional analog to the recursion process that allows efficient solutions to one-dimensional banded systems, such as tridiagonal systems of linear equations.

For use in the POP code, the solver was applied to a nine-point stencil. This is appropriate for the two-dimensional nature of the barotropic system. Periodic boundary conditions were used in the longitudinal direction. Dirichlet conditions were used for latitude. Parallelism was implemented via MPI. Domain decomposition was not changed from that used in the original POP code. Dynamic memory management was used throughout the new solver implementation. The solver had been written in C, so a C-FORTRAN interface was constructed and used for POP. The code has been tested on the Compaq cluster of machines at LLNL and on the IBM SP machines gseaborg and seaborg at NERSC.

Performance results reported here were obtained on the seaborg IBM SP machine at the NERSC facility in California. Performance results for the PCG and WFR solvers are presented in **Table 8**, and their respective initialization costs in **Table 9**. Because of a variety of difficulties, complete results have been obtained only for a relatively coarse resolution problem with 100 longitude x 116 latitude zones. The old algorithm uses nonpreconditioned PCG; the new algorithm uses WFR. Relative residual accuracy for the PCG solver was set to 1 part in 10^{13}. Accuracy for the WFR solver was at machine roundoff, typically about 1 part in 10^{15}. For the test problem, the PCG system being solved was strongly convergent (never needing more than two iterations), so the comparison has PCG performance at near-optimal conditions. During the tests, timings were taken of many other code sections as well. They remained essentially identical, indicating that the PCG-WFR comparisons are accurate. Because of the choices for zoning and domain decomposition, the cases for 32 and 64 processors are affected by load imbalance.

Table 8. Performance of preconditioned conjugate gradient (PCG, old algorithm) and wavefront recursion (WFR, new algorithm) for POP barotropic solve. Times are elapsed seconds used to update the barotropic equations for four timesteps for a 100x116 problem, omitting the cost of the initial timestep.

Processor Count	Domain Config.	PCG (old) Time	WFR (new) Time
2	1 x 2	0.27	0.36
4	2 x 2	0.17	0.17
8	2 x 4	0.13	0.16
16	4 x 4	0.12	0.22
32	4 x 8	0.15	0.49
64	8 x 8	0.18	0.86

Table 9. Initialization performance of preconditioned conjugate gradient (PCG, old algorithm) and wavefront recursion (WFR, new algorithm) for POP barotropic solve. Times are elapsed seconds used to update the barotropic equations for the initial timestep for a 100x116 problem.

Processor Count	Domain Config.	PCG (old) Time	WFR (new) Time
2	1 x 2	0.07	12.69
4	2 x 2	0.05	10.02
8	2 x 4	0.04	37.26
16	4 x 4	0.04	110.93
32	4 x 8	0.05	424.12
64	8 x 8	0.05	1133.35

The results indicate that the present implementation of the WFR algorithm is barely competitive with the PCG algorithm on a simple test problem. Additionally, the WFR algorithm has a substantial initial cost that would take hundreds or thousands of update cycles to amortize. On difficult problems, the WFR algorithm performance would be unaffected, while the PCG algorithm would be made more costly by an increased iteration count and the cost of any preconditioning. Thus, it remains an open question as to which algorithm would be preferable on a given problem. The WFR algorithm, as implemented, does not scale well on a small problem as the number of processors is increased. Two reasons for this are that the size of the reduced system increases roughly as the square of the number of processors and that the WFR implementation has each processor solve the entire reduced system (i.e., not in parallel) Obviously, a parallel solve for the reduced system is appropriate.

At the end of the effort, the decision was made not to implement the wave front recursion method as a barotropic solver in the released versions of POP. There are several reasons for this decision. The new method scaled poorly in precisely the same regime where the existing PCG method scaled poorly, namely coarse resolution problems at high processor counts. Also, the CCSM Ocean Working Group decided on a one-degree grid with 40 vertical levels and the use of two very expensive physical parameterizations. This combination of choices resulted in the barotropic solver being reduced to a very small fraction of the total simulation time and barotropic performance no longer was the critical performance issue. Third, the next release of POP will include a block distribution method which will limit the scaling penalties resulting from the barotropic solver. Finally, the transition to a hybrid vertical coordinate in the next generation POP code will include an explicit subcycling of the barotropic mode, eliminating the need for an elliptic solver.

Investigation of performance and scaling for larger test cases is under way. Configurations with 160x192 and 320x384 zones are being tested. Future work includes the use of the “SuperLU” package provided by J. W. Demmel of UC Berkeley to provide a parallel solve for the reduced system of equations.

8 Parallel I/O

The CCSM I/O system currently has two distinct components, the history files and the restart files. History files are written in netCDF format, while restart files are written using the efficient Fortran unformatted binary modes.

8.1 Performance Studies of Parallel I/O Systems

We have investigated various I/O and files systems such as netCDF, MPI-IO, GPFS, and HDF5 on IBM SP systems, to check their suitability for I/O on high performance-portable CSM. Here is a brief summary of each:

1. NetCDF. We installed netCDF on the NERSC IBM SP-3, and it runs well in sequential cases. Detailed information on netCDF performance is given in http://www.nersc.gov/research/SCG/acpi/IO/. A few highlights.
    a) Writing a netCDF file can be done at about 20-50 MB/s, depending on file sizes.
    b) Fill (or initialize allocated space) can be expensive.
    c) Implicit real*8 to real*4 conversion (used in CSM) is not efficient. Explicit conversion before a netCDF write can speed up I/O by a factor of two and reduce memory buffer by half. A critical issue is that there is currently no parallel netCDF support. NERSC is, however, developing a parallel netCDF.
2. MPI-IO. A subset of MPI-IO functionalities is provided on the NERSC IBM SP-3 that gives reasonable I/O rates with the right data distribution. The data reshuffle from the computational XZY index order to the required XYZ order by netCDF, however, can be slow by orders of magnitude. It must be optimized by programmers. We are investigating MPI-IO's potential use for I/O restart.
3. HDF5. We are interested in HDF primarily because of its parallel I/O support for netCDF. A Fortran interface is in the development stage. One disadvantage of HDF5 is that users have very limited experience with it.
4. GPFS (IBM's native parallel file system). A comprehensive performance analysis has been done. The basic point is that performance varies on different processors and file sizes. To write out data larger than 32 MB, GPFS can achieve a bandwidth of 100 MB/sec. See the Web page http://www.nersc.gov/research/SCG/acpi/sp_io.html

8.2 Improving CCSM I/O Performance

We report on two specific achievements relating to CCSM:

- CCM history I/O improvements. We developed a flexible mechanism for setting the precisions (single or double) of the history buffer and netCDF file selectively for individual fields, and we incorporated this into the current CCM. The mechanism allows substantial memory savings and also speedup in I/O for those fields that do not require double precision. This work is prompted by our earlier netCDF performance study that implicit precision conversion should be avoided. Changes to the CCM code suites are checked and validated.
- I/O library and utility requirements. We completed a requirements analysis. See the Web page http://www.scd.ucar.edu/css/infrastructure/history/design_docrequirements.html

Based on this work, we have initiated a project to develop a parallel I/O library for CCSM. We are in the process of designing a set of low-level parallel I/O functions that are portable and efficient. They can be used in several component models, and for different file format, netCDF, Fortran unformatted, *et cetera*. We are studying the interface with CCM and POP. Many issues (such as packing different fields, history buffer reorganization, data transpose between different decompositions) are being carefully examined.

9 Code Development Methodologies and Infrastructure

The success of this project depended critically on the coordination of the many scientists and software engineers from DOE labs, NCAR, and NASA. From the outset, we adopted the philosophy that this coordination required a clearly defined and effective software development process, so that (1) the improvements described in this proposal were actually incorporated into the CCSM code, and (2) the CSM group assumed maintenance and future responsibilities for the code. The elucidation of a clear and practical development process that can be merged with the CSM group's current development practice was an important deliverable of this project.

The five basic elements of our development strategy were as follows:

1. The development of a CSM Software Developers' Guide, which describes a clear set of coding practices and standards.
2. The development of a series of design documents, which articulate both overall code structure and the design of specific components.
3. The definition of a jointly agreed upon staged software development cycle that includes formal technical reviews (FTRs) and quality assurance (QA) procedures.
4. The use of a common code repository.
5. Communication and archival mechanisms that will keep the many developers involved in the project aware of what has been done and what needs to be done.

We believe that we were largely successful in establishing these processes and that our success in this area represents a significant step forward for joint DOE-NSF research and development work in the climate area.

9.1 CSM Software Developers' Guide

Working with the CCSM Software Engineering Working Group, the Avant Garde team under the lead of Brian Kauffman developed a *CCSM Software Developers' Guide* that explains and documents key software practices relating to the CCSM. These include the recommended software development cycle, expected documentation, configuration management procedures, coding standards, and testing and validation procedures. A first release of the guide was distributed at the 6th Annual CCSM Workshop and is available at http://www.ccsm.ucar.edu/models.

9.2 Design Documents

The logical first step in the software development methodology we adopted is preparing a statement of what the software must do. This *requirements* document was then analyzed and used as the basis for a software design. A requirements document may include scientific requirements, but for the case of the atmsopheric model, we restricted attention to functional requirements. The second document we produced was an *architecture description* that explains in broad terms how the requirements will be met. This document is very useful for developers to know how the code structure will change to meet the requirements.

Document templates were developed to give a uniformity of structure to the designs of each component. These templates also make it possible to reuse document parts and automate the procedure of producing hardcopy, html, and pdf versions from the same text base. The base document is maintained in LaTeX.

The design documents communicate the desired high-level structure for CSM and CCM. These documents include a description of major data structures and high level calling tree.

Specific design documents were prepared for individual code components such as the interface to the physical parameterization package in the atmospheric model. The level of detail in these interface descriptions is not suitable for high-level documents and also must be kept up to date with the actual state

of the code. To facilitate the updating of the interface design documents, we adopted the ProTEX system developed by NASA. Based on structured comments in the code itself, the ProTEX tool produces a formatted description of subroutine and module interfaces. These descriptions are then incorporated in the design documents where appropriate. Thus, the code is self-documenting to a limited extent. The design documents listed in the appendix are the product of these tools.

9.3 Staged Software Engineering Development Cycle

We proposed to introduce a staged (or iterative) development cycle as our software engineering methodology for this project. This was achieved to a large extent, though other constraints forced us to slowly adapt to this methodology. Each software component (coupler, I/O library, dynamical cores) stepped through the following stages.

1. *Write and review a requirements list*.

This was done for both the atmosphere and coupler with good success.

2. *Write and review a design document*.

The development of architecture documents was accompanied by the creation and evaluation of code prototypes. We found these documents useful as living documents available on the web and easily updated to reflect the present thinking and discussions. Only as the code matures and the design flaws are wrung out has it been appropriate to finalize and review these documents.

3. *Write and review code*.

Code was written in accordance with standards laid out in the Developers' Guide. The code review consisted of a small number of developers who were following each other's work. A more formal procedure carefully inspecting the code should be done before release but was not done adequately in the Avant Garde project. Adopting a more formal procedure would be an excellent way to identify mistakes and inefficiencies at an early stage of development. Code building was automated through scripts on all the platforms of interest

4. *Unit test code (test stand-alone).*

Testing of code before checking into the repository a high priority part of the development process. On a few occasions, the testing did not catch mistakes. The testing procedure was amended several times during this project to be more comprehensive and reliable.

5. *Integrate and validate code.*

Incorporating a code segment into a working version of the CCM and ensuring that it correctly interoperates with other portions of the code were the responsibility of the developer. All developers had write access to the code repository trunk. This was not the case with the coupler development team because many of the developments represent major restructuring.

Many of the changes we made to the CSM code involved optimization and reorganization of the source code and thus produced differences in numerical results that are only of round-off magnitude or, no change at all. Given an appropriate definition of "round-off," testing of these changes could be automated. No profound algorithmic changes, such as a new solver for the primitive equations or new physics parameterizations, were made in the Avant Garde project and would require additional testing to ensure the model will run well over long-term integrations. This level of testing is called validation.

6. *Update the design document, and convert it to documentation.*

The Avant Garde project completed one major iteration of this cycle for the coupler and atmospheric components.

9.4 Formal Technical Reviews

The requirements documents, coding style manuals, and design documents were subject to a formal review process. The code review process at NCAR currently only applies to a cleanup before release. NCAR has always had scientists and developers review code before a public release. As is well known and recognized throughout the climate community, the past versions of the CCM exhibit excellent coding style and

craftsmanship. Maintaining this tradition of quality is a serious challenge now that the development group has expanded.

The fact that the CCSM is a community model—and that the model will be used in an operational forecast and assessment production setting—provides motivation for careful software quality assurance. The approach to peer review through formal technical reviews or walkthroughs (McConnell 1993) mentioned in the preceding section is a procedure that would benefit CCSM development. A review or walkthrough is a moderated meeting in which requirements, software design, or a piece of code is examined and discussed.

The three types of reviews have different participants:

- scientists, who review requirements
- scientists and software developers who review the design document, or
- developers who review code.

Possible outcomes of a review are to:

- schedule a date for release of a final version incorporating reviewer comments or
- schedule a date for another review.

Though the Avant Garde did not achieve the desired level of software review, the CCSM now has a software coordinator and is in a position to implement these broader practices in follow-on projects.

9.5 Developers' Repository

Since scientists and software engineers from both NCAR and DOE labs have experience with the Concurrent Versions System (CVS), we used it as the basis for a common repository and version control. From a management point of view, CVS is desirable because there are both Windows and Macintosh implementations that include a graphical user interface (see http://www.vincvs.org). CVS is also free software, which makes it appealing to university researchers. It is downloadable by anonymous ftp at http://www.sourcegear.com/CVS.

Erik Kluzek developed a script that runs on check in to the CVS repository which requires that the developer answer a standard set of questions regarding the modifications being committed. The form is automatically archived (with the code) and e-mailed to the developers mailing list. Each day there are several commits and this mechanism was helpful in keeping up-to-date. The daily updates and testing that the CVS system facilitates on remote (to NCAR) systems was very important to the success of the Avant Garde project.

The decision to grant DOE and NASA programmers access to the CCSM code repository created coordination issues for the project and for NCAR but was essential for the progress of our work. The atmospheric team followed a full inclusion – full responsibility model while the coupler and ocean teams pursued a component development and prototype model coordinated with repository managers. Both approaches were successful and were taken for valid reasons. The atmospheric model development was required to always provide scientifically usable code (i.e. no down time for model restructuring). This required that code modifications by our project be staged and coordinated with other working groups using the developmental version. The next-generation coupler development was such a great departure from the CSM-1 code that a more independent development was possible.

The problem with repositories that we did not solve was how to protect intellectual property on development branches. CVS does not support selective or exclusive access in its checkout procedures. This implies that all developments are viewable by anyone with read privileges. For scientific work that is yet to be published using the CCSM model the lack of protections on check-out, has proved to be a barrier to CCSM model developers. Our project spent some effort evaluating possible fixes and alternative version control systems. Unfortunately, the other systems each had significant flaws, and CVS still appeared to be the best tool for our purposes. At one point, we considered hiring a systems programmer to modify the CVS system to our specifications. A person was identified that understood the problem and had experience with the CVS source code. But there was no arrangement we were able to make that would have moved our investment into the standard CVS distribution. Thus future versions would need to be modified again. Hoping that we could cope for the short term of this project, we chose not to make the modifications to CVS. This issue will require some action by future development teams and CCSM management.

9.6 Communication Mechanisms

Several mechanisms for keeping developers apprised of the current state of the project and its future directions were developed and utilized by the project. The CCSM Software Engineering Working Group Web page posted the public documents, while a project Web page included task lists and draft documents. Several e-mail reflectors and archive lists were maintained by the project. Through the duration of the project a weekly teleconference was held. In addition, the coupler and atmospheric task groups met during the project for intense coding or work on documents. Regular project meetings were held, rotating among NCAR and the DOE laboratory sites. Notes from these meetings were posted on the Web. Finally, the Breckenridge CCSM Workshop in June served as a meeting point for the project. We also mention that the CCSM Software Engineering Working Group typically held a meeting on the first morning of our *Avant Garde* project meetings. From a practical point of view, this made sense because of the large intersection of people involved with *Avant Garde* and the SEWG., but it also served to enhance the coordination on software issues.

10 Summary and Future Directions

The goals of the *Avant Garde* pilot project have been largely achieved. Throughput and scalability issues within the CCSM have been addressed by introduction of new parallel decompositions and generalized coupling mechanisms that reduce or eliminate the communication bottlenecks present in the CSM-1 design. Extensibility has been addressed by a redesign of the software into a modular, layered structure that will improved code maintainability as well as facilitate future developments by distributed development teams. A next-generation coupler has been produced, and the atmospheric code shows enhanced performance and scalability in the physics and finite volume dynamical core. The ability of the coupled model to run effectively on multiple platforms has been significantly enhanced by the extension of the development teams to several DOE sites and by the use of formal testing procedures (including multiplatform testing) in the software development process. The PCM and CSM development tracks are closer to a merged product with the development of key components that meet the design requirements set forth at the beginning of the project.

Several critical decisions helped (and some hindered) the productivity of our development team. The decision to grant DOE and NASA programmers access to the CCSM code repository created coordination issues for the project and for NCAR but was essential for the progress of our work. The decision to pursue an object-oriented style using Fortran90 positioned the CCSM development for good computational performance as well as the option to replace entire libraries, levels, or components with other language instantiations. The Fortran90 hybrid distributed /shared-memory programming paradigm that we adopted provided good support for the exploitation of present and future computer architectures and provided a transition strategy for many of the development team that were previously familiar with only one or the other style of programming in a parallel environment. The object-oriented approach also allowed us to borrow many of the practices of software development projects from nonscientific domains. The transition from Fortran 77 (with common blocks) to Fortran90 modules also occurred during the eighteen months of this project. In all, major code restructuring has been performed in this project, and much has been accomplished, but there is still much to do.

Future projects will extend the pilot project, by including all of the components of the CCSM and broadening the scope to include model development. The software design effort will expand to the ocean, sea ice, and land-surface models. The software design must also accommodate the requirements of model development in chemistry and biogeochemistry that were not present in the *Avant Garde* project but that are of great relevance to energy research. Requirement and architecture design documents and implementation plans will be developed, in conjunction with the CCSM Software Engineering Working Group, to bring all components into conformity with a comprehensive software design including a machine-specific layer, a utility and library layer, and a model-code layer. This software hierarchy will enable rapid adaptation to new architectures while maintaining high production efficiencies on the available computing platforms and complete the direction established in the *Avant Garde* project. The modular structure will make the substitution of alternative components significantly easier than in current models, and in future work we anticipate that many projects will use the CCSM as a code base for their research.

The limited time of the *Avant Garde* project precluded investigating the development of new or improved model formulations, numerical algorithms, and parameterizations of physical and chemical processes; extension of component models to higher resolutions; and diagnosis, evaluation and optimization of model performance, both scientific and computational. Future work will require a broader involvement with the model development to make the CCSM the best possible.

The goal of future work and follow on projects should be to collaboratively develop CCSM so that it (1) is comprehensive in its treatment of physical and chemical processes important to the climate system; (2) comprises modular packages with well-defined interfaces that can be tested off-line and interchanged with packages containing updated or alternative treatments of the same processes, and (3) is performance optimized yet portable and adaptable for new computing architectures.

A DOE SciDAC project involving an expanded list of participants from DOE labs, NCAR, and NASA has been awarded for FY02.

Acknowledgements

The authors wish to thank many people for their contributions to the success of this project. We thank: Will Sawyer and S.-J. Lin of the NASA Data Assimilation Office, and Jim Rosinski and Erik Kluziek of NCAR's Climate and Global Dynamics Division for their contributions to the atmosphere model portion of the project; Mariana Vertenstein of NCAR's Climate and Global Dynamics Division, and Richard Loft, Rodney James, and Dan Anderson of NCAR's Scientific Computing Division for their contributions to the requirements and design processes for the next-generation coupler; Jace Mogill and Celeste Cory of Cray Research for their advice, bug fixes, and performance enhancements to the Model Coupling Toolkit; John Michalakes of NCAR's Mesoscale and Microscale Meteorology Division for the useful and illuminating discussions throughout this project; and Gail Peiper of Argonne National Laboratory's Mathematics and Computer Science Division for her extensive proofreading and many corrections and useful suggestions for improving the quality of this report. The ACPI *Avant Garde* project was funded by the United States Department of Energy Office of Biological and Environmental Research under Field Work Proposal 66204, KP1201020.

Appendix. Project Document Web Links

The achievements of the *Avant Garde* project are further documented in technical reports, papers and working documents listed here. Many of these appear in draft form or are in the review process.

The CCSM Software Engineering Working Group has written the following documents guiding software development. Several of the authors and reviewers on these documents were DOE supported by the ACPI *Avant Garde* Project but also reflect the larger CCSM effort.

CCSM Software Engineering Plan for 2000-2005:
(http://www.ccsm.ucar.edu/csm/working_groups/Software/plan2000-2005).

CCSM Software Developers' Guide
(http://www.ccsm.ucar.edu/csm/working_groups/Software/dev_guide/dev_guide)

Atmosphere Model Document Web Links

1. Requirements
   - **a.** CAM Software Requirements
     (http://www.cgd.ucar.edu/csm/models/atm-cam/docs/atm_reqdoc).
2. Design
   - **a.** CAM Software Architecture
     (http://www.cgd.ucar.edu/csm/models/atm-cam/docs/atm_archdoc).
   - **b.** Interface to Column Physics and Chemistry Packages
     (http://www.cgd.ucar.edu/csm/models/atm-cam/docs/phys-interface)

Coupler Document Web Links

1. Requirements
   a. Next Generation Coupler Requirements Document
   (http://www.cgd.ucar.edu/csm/models/cpl6/docs/cpl6_reqdoc/cpl6_reqdoc.html)
2. Design
   a. MPH: A Library for Distributed Multi-Component Environment
   (http://www.nersc.gov/research/SCG/acpi/MPH/mph_doc)
   b. CCSM Coupler Architecture – CPL6
   http://www.cgd.ucar.edu/csm/models/cpl6/docs/cpl6_archdoc/cpl6_archdoc.html
   c. The Model Coupling Toolkit API Definition Document
   http://www.mcs.anl.gov/acpi/mct/mct_APIs.pdf

Publications

The following publications report on work partially or wholly supported by the Avant Garde project:

MPH: a Library for Distributed Multi-Component Environment, Chris Ding and Yun He. April 2001. LBNL Tech Report 47929

A Ghost Cell Expansion Method for Reducing Communications in Solving PDE Problems. Chris Ding and Yun He. March 2001. Accepted by SC 2001.

Using Accurate Arithmetics to Improve Numerical Reproducibility and Stability in Parallel Applications. Yun He and Chris Ding. Journal of Supercomputing ,Volume 18, Issue 3, March 2001, pp.259-277.

An Optimal Index Reshuffle Algorithm for Multidimensional Arrays and its Applications for Parallel Architectures. Chris Ding. IEEE Transactions on Parallel and Distributed Systems, March 2001, v.12, pp.306-315.

The Model Coupling Toolkit, J.W. Larson, R.L. Jacob, I.T. Foster, and J. Guo, in *Proceedings of the International Conference on Computational Science (ICCS)* 2001, V.N. Alexandrov, J.J. Dongarra, B.A. Juliano, R.S. Renner, and C.J.K. Tan (eds.), Springer-Verlag Lecture Notes in Comupter Science Volume 2073, pp 185-194 (2001) **(Also available as ANL/CGC-007-0401)**

The Computational Complexity, Parallel Scalability, and Performance of Atmospheric Data Assimilation Algorithms, P.M. Lyster, J. Guo, T. Clune, and J.W. Larson, Submitted to *Computers in Science and Engineering*. Available on-line at the URL ftp://dao.gsfc.nasa.gov/pub/papers/lyster/submitted.pdf

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, to appear in *Proceedings of the 1999 High Performance Distributed Computing Conference*.

Ashworth, M. 1999. Optimisation for vector and RISC processors, in *Towards Teracomputing*, World Scientific, River Edge, NJ. pp. 353-359.

Bitz, C. M., Holland, M. M., Eby, M. and Weaver, A. J. 2000. Simulating the ice thickness distribution in a coupled climate model. preprint.

Bitz, C. M. and Lipscomb, W. H. 1999. An energy-conserving thermodynamic model of sea ice. *J. Geophys. Res*. **104**, 15669-15677.

Bonan, G. 1998. The land surface climatology of the NCAR land surface model coupled to the NCAR community climate model, *J. Climate*, **11**, 1307-1326.

Briegleb, B., and Bromwich, D. H. 1998. Polar radiation budgets of the NCAR CCM-3, *J. Climate*, **11**, 1246-1269.

Brooks, Frederick P., Jr. 1995. *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition*, Addison-Wesley, Reading, Massachusetts.

Climate System Model 1998. Special issue, *J. Climate* **11,** no. 6.

Colella, P., and Woodward, P. 1984. The piecewise parabolic method (PPM) for gas-dynamical simulations*, J. Comput. Phy*s, **54**, 174-201.

DAO 2000: Algorithm Theoretical Basis Document, Version 2.0, Data Assimilation Office, NASA Goddard Space Flight Center, Greenbelt, Maryland 20771. Available online at http://dao.gsfc.nasa.gov/subpages/atbd.html.

Ding, C. H. Q., and He, Y. 1999. Data organization and I/O in a parallel ocean circulation model, *Proc. SC99*.

Drake, J., Foster, I., Michalakes, J., Toonen, B., and Worley, P. 1995. Design and performance of a scalable parallel community climate model, *Parallel Computing*, **21**, 1571-1592.

Dukowicz, J. K., and R. D. Smith. 1994. Implicit free-surface method for the Bryan-Cox-Semtner ocean model, *J. Geophys. Res*. **99,** 7991-8014.

Dukowicz, J. K., Smith, R. D., and Malone, R. C. 1993. A reformulation and implmentation of the Bryan-Cox-Semter ocean model on the Connection Machine, *Atmos. Ocean Tech*. **10,** 195-208.

Flato, G., and Hibler, W. 1992. Modeling pack ice as a cavitating fluid, *J. Phys. Oceanogr*., **22**, 636-651.

Gates, W. L. et al. 1996, Climate Models - Evaluation, Chapter 5 of the Intergovernmental Panel on Climate Change Second Scientific Assessment of Climate Change, CUP, pp. 233-276.

Gent, P. R., and McWilliams, J. C. 1990. Isopycnal mixing in ocean circulation models, *J. Phys. Oceanography* **20,** 150-155.

Hack, J. 1994. Parameterization of moist convection in the NCAR community climate model*, J. Geophys. Res*., **99**, 5541-5568.

Hack, J., Kiehl, J., and Hurrell, J. 1998. The hydrologic and thermodynamic characteristics of the NCAR CCM-3, *J. Climate*, **11**, 1179-1206.

Holtslag, A., and Boville, B. 1993. Local versus nonlocal boundary-layer diffusion in a global climate model, *J. Climate*, **6**, 1825-1842.

Hunke, E., and Dukowicz, J. 1997. An elastic-viscous-plastic model for sea ice dynamics, J. *Phys. Oceanogr.*, **27**, 1849-1867.

Hunke, E. C., and Lipscomb, W. H. 1999. CICE: The Los Alamos sea ice model, documentation and software, Version 2, Los Alamos National Laboratory, LA-CC-98-16 v.2,

Hurrell, J., Hack, J., and Boville, B. 1998. The dynamical simulation of the NCAR community climate model CCM-3, *J. Climate*, **11**, 1207-1236.

Jones, P. W. 1999. First- and second-order conservative remapping schemes for grids in spherical coordinates, *Monthly Weather Rev*. **127,** 2204-2210.

Kattenberg, A., Giorgi, F., Grassl, H, Meehl, G. A., Mitchell, J. F. B., Stouffer, R. J., Tokioka, T., Weaver, A. J., and Wigley, T. M. H. 1996: Climate Models - projections of future climate, in *Climate Change 1995 the Science of Climate Change The Second Assessment Report of the IPCC: Contribution of Working Group I* (Eds. J. T. Houghton, L. G. Meira Filho, B. A. Callander, N. Harris, A. Kattenberg, and A. Maskell), Cambridge University Press, 285-357.

Kiehl, J., Hack, J., Bonan, G., Boville, B., Williamson, D., and Rasch, P. 1998a. The National Center for Atmospheric Research community climate model: CCM-3, *J. Climate*, **11**, 1131-1149.

Kiehl, J., Hack, J., and Hurrell, J. 1998b. The energy budget of the NCAR community climate model CCM-3, *J. Climate*, **11**, 1151-1178.

Large, W. G., McWilliams, J. C., and Doney., S. C. 1994. Oceanic vertical mixing: A review and a model with a non-local boundary layer parameterization, *Rev. Geophysics* **32,** 363-403.

Lin, S.-J. 1997. A finite-volume integration method for computing pressure gradient force in general vertical coordinates, *Quart. J. Roy. Meteor. Soc.* **123**, 1749-1762.

Lin, S.-J and Rood, R. B. 1996. Multidimensional flux-form semi-Lagrangian transport schemes*, Mon. Wea. Rev*. **124**, 2046-2070.

Lin, S.-J and Rood, R. B. 1997. An explicit flux-form semi-Lagrangian shallow water model on the sphere, *Quart. J. Roy. Meteor. Soc.* **123**, 2477-2498.

Lin, S.-J., and Rood, R. B. 1998. A flux-form semi-Lagrangian general circulation model with a Lagrangian control-volume vertical coordinate. The Rossby-100 symposium, Stockholm, Sweden.

Lin, S.-J., and Rood., R. B. 1999. Development of the joint NASA/NCAR General Circulation Mode. Preprint, 13th conference on Numerical Weather Prediction, Denver, CO.

McConnell, S. 1993. *Code Complete, a Practical Handbook of Softwware Construction*, Microsoft Press, Redmond, Washington.

Meehl, G. A., Gent, P., Arblaster, J., Otto-Bliesner, B. Brady, E., and Craig, A. 2000. Factors that affect amplitude of El Niño in global coupled climate models. In preparation for Climate Dynamics.

Michalakes, J. 2000. The Same-Source Parallel MM5. *Journal of Scientific Programming*, **8**, 5-12.

Michalakes, J., Chen, S., Dudhia, J., Hart, L., Klemp, J., Middlecoff, J., and Skamarock, W. 2001. "Development of a Next Generation Regional Weather Research and Forecast Model" in *Developments in Teracomputing*: Proceedings of the Ninth ECMWF Workshop on the Use of High Performance Computing in Meteorology. Eds. Walter Zwieflhofer and Norbert Kreitz. World Scientific, Singapore. pp. 269-276

Michalakes, G., Dudhia, J., Gill, D., Klemp, J., and Skamarock, W. 1998. Design of a next-generation weather research and forecast model, in *Proceedings of the Eighth Workshop on the Use of Parallel Processors in Meteorology*, European Center for Medium Range Weather Forecasting, Reading, U.K., November 16-20, 1998. Available as ANL/MCS preprint number ANL/MCS-P735-1198.

Rosinski, J. M. and Williamson, D. L. 1997: The accumulation of the rounding errors and port validation for global atmospheric models. *SIAM J. Sci. Comput*. **18**, 552-564.

Semtner A., 2000. Ocean and climate modeling on advanced parallel computers: progress and prospects, *Communications of the ACM* (in press).

Smith, R. D., Dukowicz, J. K., and Malone, R. C. 1992. Parallel ocean general circulation modeling, *Physica* **D 60,** 38-61.

Smith, R. D., Kortas, S., and Meltz, B. 1995. Curvilinear coordinates for global ocean models, Los Alamos National Laboratory Report LAUR-95-1146.

Sterling, T., Messina, P., and Smith , P. H. 1995. *Enabling Technologies for PetaFLOPS Computing,* MIT Press, Cambridge, Massachusetts.

Van Leer, B. 1977. Toward the ultimate conservative difference scheme, Part IV: A new approach to numerical convection. *J. Comput Phys*., **23**, 276-299.

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

Washington, W. M., Weatherly, J. W., Meehl, G. A., Semtner, A. J., Bettge, T. W., Craig, A. P., Strand, W. G., Jr., Arblaster, J., Wayland, V. B., James, R., and Zhang, Y. 2000. Parallel climate model control and transient simulations. *Climate Dynamics* (in press).

Weatherly, J. W. and Zhang, Y. 2000. The response of the polar climate to increasing CO_2 in a global climate model with elastic-viscous-plastic sea ice, *Journal of Climate* (accepted).

Winton, M. 1998. A reformulated three-layer sea ice model, preprint

Williamson, D.L., Bath, L. M., Sato, R. K., Mayer, T. A., and Kuhn, M. L. 1983: Documentation of NCAR CCM0B program modules. NCAR Technical Note NCAR/TN-212+IA, Boulder, Colorado, NTIS No. PB83 263996

Williamson, D. L., and Rosinski, J. M. 2000: Accuracy of reduced grid calculations, *QJRMS* (in press).

Zhang, G., and McFarlane, N. 1995. Sensitivity of climate simulations to the parameterization of cumulus convectionin the Canadian Climate Centre general circulation model, *Atmos.-Ocean*, **33,** 407-446.