



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

The Implementation of the Finite-Volume Dynamical Core in the Community Atmosphere Model

W. B. Sawyer, A. A. Mirin

July 27, 2005

First Indo-German Conference on PDE, Scientific Computing
and Optimization in Applications
Trier, Germany
September 8, 2004 through September 10, 2004

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

The Implementation of the Finite-Volume Dynamical Core in the Community Atmosphere Model

William B. Sawyer^{a,b}

^a*Swiss Federal Institute of Technology (ETHZ), Seminar for Applied Mathematics
Rämistrasse 101, Zürich, 8092 Switzerland*

^b*Global Modeling and Assimilation Office (GMAO), Goddard Space Flight Center
Greenbelt, MD 20771 USA*

Arthur A. Mirin^c

^c*Lawrence Livermore National Laboratory, Livermore, California, 94550 USA*

Abstract

A distributed memory message-passing parallel implementation of a finite-volume discretization of the primitive equations in the Community Atmosphere Model 3.0 is presented. These three-dimensional equations can be decoupled into a set of two-dimensional equations by the introduction of a floating vertical coordinate, resulting in considerable potential parallelism. Subsequent analysis of the data dependencies — in particular those arising from the polar singularity of the latitude-longitude coordinate system — suggests that two separate domain decompositions should be employed, each tailored for a different part of the model. The implementation requires that data be periodically redistributed between these two decompositions. Furthermore, data from nearest neighbors are kept in halo regions, which are updated between iterations. These data movements are optimized through one-sided communication primitives and multithreading. The resulting algorithm is shown to scale to very large machine configurations, even for relatively coarse resolutions.

Key words: Atmospheric Dynamics, Primitive Equations, Finite-Volume Methods, Parallel Computing

PACS: 92.60.Bh

1 Introduction

Atmospheric general circulation models (AGCMs) are key tools for weather prediction and climate research. They also require large computing resources:

even the largest current supercomputers cannot keep pace with the desired increases in the resolution of these models. AGCMs consist, roughly speaking, of the *dynamics*, which calculates the atmospheric flow, and the *physics*, in which parameterizations for subgrid phenomena such as long- and short-wave radiation, moist processes, and gravity wave drag, are approximated. The physical parameterizations will not be discussed further here. We concentrate on the finite-volume (FV) solver of the dynamics (the *dynamical core*) which requires a substantial fraction of the overall computational time of the Community Atmosphere Model (CAM 3.0, part of the Community Climate System Model [2]).

The mathematical formulation of the primitive equations is presented in Sec. 2, and the finite-volume discretization in Sec. 3. An analysis of the data dependencies of the scheme and their implications for the domain decomposition on parallel computers are given in Sec. 4. In Sec. 5, an overview of the parallel implementation is presented. Results are presented in Sec. 6, where it is seen that the approach scales well to very large machine configurations. Additional conclusions and future directions are presented in Sec. 7.

2 Mathematical Formulation

We consider an atmosphere in hydrostatic balance:

$$\frac{1}{\rho} \frac{\partial p}{\partial z} = -g, \quad (1)$$

where ρ is the density, p the pressure, and g the force per unit mass due to gravity. This physically justified assumption is part of the primitive equations describing atmospheric motion, of which the conservation of mass and momentum are the most relevant here:

$$\frac{\partial \rho}{\partial t} = -\vec{\nabla}_3 \cdot (\rho \vec{v}), \quad (2)$$

$$\frac{d\vec{v}}{dt} = -\frac{1}{\rho} \vec{\nabla}_3 p - \vec{\nabla}_3 \Phi + \vec{F} - 2\vec{\Omega} \times \vec{v}, \quad (3)$$

where Φ is the geopotential, \vec{F} is the frictional force (which will not be considered further in this work), and $\vec{\Omega}$ is the earth's angular velocity. We use $\vec{\nabla}_3$ to explicitly designate the three-dimensional gradient operator to avoid confusion with the horizontal gradient which will be used subsequently. The full set of the primitive equations can be found in [6]. In this work, only the

horizontal wind components will be considered. In this case, the Coriolis term $2\vec{\Omega} \times \vec{v}$ is replaced by its horizontal component, $f \approx 2\Omega \sin \theta$, where θ is the latitude.

In place of z , a general vertical coordinate ζ can be introduced if a bijective transformation exists:

$$z \xleftrightarrow[T, T^{-1}]{} \zeta. \quad (4)$$

That is to say, ζ is a monotone function of z . There are a number of sensible choices for ζ (see [6]), for example, ζ can represent isentropic or terrain-following coordinates. The gradient of a scalar quantity $Q(x, y, \zeta)$ can be written as,

$$\vec{\nabla}_\zeta Q = \vec{\nabla}_z Q + \frac{\partial Q}{\partial \zeta} \frac{\partial \zeta}{\partial z} \vec{\nabla}_\zeta z, \quad (5)$$

where the subscript of $\vec{\nabla}$ indicates which horizontal surface, ζ or z , remains constant in the vertical. In order to formulate Eq. (2) in ζ , a pseudo-density π can be defined,

$$\pi = -\frac{\partial \Phi}{\partial \zeta} \rho. \quad (6)$$

By making use of Eq. (1), it follows that $\pi = \partial p / \partial \zeta$. The mass continuity and horizontal momentum equations in ζ coordinates become,

$$\frac{d\pi}{dt} + \pi \vec{\nabla}_\zeta \cdot \vec{v}_H + \pi \frac{\partial \dot{\zeta}}{\partial \zeta} = 0, \quad (7)$$

$$\frac{d\vec{v}_H}{dt} + \frac{1}{\rho} \vec{\nabla}_\zeta p + \vec{\nabla}_\zeta \Phi - \vec{F} + f \vec{k} \times \vec{v}_H = 0, \quad (8)$$

where $\dot{\zeta}$ is the time derivative of ζ . By making use of the relationship,

$$\frac{d\pi}{dt} = \frac{\partial \pi}{\partial t} + \vec{v}_H \cdot \vec{\nabla}_\zeta \pi + \dot{\zeta} \frac{\partial \pi}{\partial \zeta}, \quad (9)$$

it follows that the conservation of total air mass and the mass of an arbitrary set of tracers (such as water vapor) can then be succinctly formulated,

$$\frac{\partial \pi}{\partial t} + \vec{\nabla}_\zeta \cdot (\vec{v} \pi) = 0, \quad (10)$$

$$\frac{\partial \pi q_i}{\partial t} + \vec{\nabla}_\zeta \cdot (\vec{v} \pi q_i) = 0, \quad (11)$$

where q_i is the mixing ratio of the tracer (i). In a Lagrangian frame — in which a control volume glides between constant ζ surfaces — the operator $\vec{\nabla}_\zeta$ is two-dimensional.

Additionally, if potential temperature is taken as the thermodynamic variable,

$$\Theta = T \left(\frac{p_{\text{ref}}}{p} \right)^{R/c_p}, \quad (12)$$

where T is the temperature, $p_{\text{ref}} = 1000$ hPa is a reference pressure, R is the gas constant for dry air, and c_p is the specific heat of dry air at a constant pressure, it follows that,

$$\frac{\partial \pi \Theta}{\partial t} + \vec{\nabla}_\zeta \cdot (\vec{v} \pi \Theta) = 0. \quad (13)$$

In the absence of friction \vec{F} , Eq. (8) can be written in latitude-longitude coordinates in its vector-invariant form with spherical coordinates [1]:

$$\frac{\partial u}{\partial t} = \eta v - \frac{1}{R_e \cos \theta} \left[\frac{\partial}{\partial \lambda} (\kappa + \Phi - \nu D) + \frac{1}{\rho} \frac{\partial p}{\partial \lambda} \right] - \frac{d\zeta}{dt} \frac{\partial u}{\partial \zeta}, \quad (14)$$

$$\frac{\partial v}{\partial t} = -\eta u - \frac{1}{R_e} \left[\frac{\partial}{\partial \theta} (\kappa + \Phi - \nu D) + \frac{1}{\rho} \frac{\partial p}{\partial \theta} \right] - \frac{d\zeta}{dt} \frac{\partial v}{\partial \zeta}, \quad (15)$$

where R_e is the earth's radius, $\eta = 2\Omega \sin \theta + \left[\frac{\partial}{\partial \lambda} v - \frac{\partial}{\partial \theta} (u \cos \theta) \right] / R_e \cos \theta$ is the absolute vorticity, $\kappa = (u^2 + v^2)/2$ is the kinetic energy, D is the divergence and ν is a coefficient which takes subgrid processes into account.

3 Finite-Volume Discretization

A finite-volume approach was proposed in [10] in which the cell-averaged values of a physical quantity \bar{Q} are defined as,

$$\bar{Q}_{i,j} = \frac{1}{A_{i,j}} \int_{\lambda_i - \Delta\lambda/2}^{\lambda_i + \Delta\lambda/2} \int_{\theta_j - \Delta\theta/2}^{\theta_j + \Delta\theta/2} Q(t; \lambda, \theta) d\theta d\lambda, \quad (16)$$

and can be updated by the upwind integral,

$$\bar{Q}_{i,j}^{n+1} = \bar{Q}_{i,j}^n - \frac{1}{A_{i,j}} \int_t^{t+\Delta t} \oint Q(\tau; \lambda, \theta) \vec{v} \cdot \vec{n} dl d\tau. \quad (17)$$

$A_{i,j}$ is the area of finite volume (i, j) . In this work, we consider only constant interval lengths $\Delta\lambda$ and $\Delta\theta$ and thus, $A_{i,j} = R_e^2 \Delta\theta \Delta\lambda \cos\theta_j$.

A number of numerical approximations for Eq. (17) were proposed in [10]. The main discretization scheme split the 2-D flux integral into two orthogonal flux-form transport operators:

$$F(u^*, \Delta t, \bar{Q}) = -\frac{\Delta t}{R_e \Delta\lambda \cos\theta} \delta_\lambda [\mathcal{X}(u^*, \Delta t; Q)], \quad (18)$$

$$G(v^*, \Delta t, \bar{Q}) = -\frac{\Delta t}{R_e \Delta\theta} \delta_\theta [\mathcal{Y}(v^*, \Delta t; Q)], \quad (19)$$

where δ is a difference operator on the argument values taken at neighboring gridpoints, and (u^*, v^*) is an approximation of time-averaged values of \vec{v} across a cell walls. \mathcal{X} and \mathcal{Y} are the time-accumulated flux operators,

$$\mathcal{X}(u, \Delta t; Q) = \int_t^{t+\Delta t} u Q d\tau, \quad (20)$$

$$\mathcal{Y}(v, \Delta t; Q) = \int_t^{t+\Delta t} v Q d\tau. \quad (21)$$

For the reconstruction of Q from \bar{Q} , the Piecewise Parabolic Method [3] is generally used. The PPM method imposes a monotonicity constraint on the 1-D discrete solution, though it does not ensure monotonicity in the overall 2-D solution. In [8] a Lagrangian frame was suggested in which ζ remains constant for some period of time after which it is remapped to a fixed vertical coordinate. In the Lagrangian frame, Eqs. (10), (11), (14) and (15) can be decoupled into a set of two-dimensional equations. An explicit, oscillation-free finite-volume method to determine the time evolution of the prognostic variables — the pressure change per vertical level Δp , the potential temperature Θ , the velocity u, v and the tracer mixing ratios q_i — was formulated in [7]. For example, the time step increments for the velocities are,

$$\Delta u = \Delta t \left[\mathcal{Y}(v^*, \Delta t; \eta^\lambda) - \frac{1}{R_e \Delta\lambda \cos\theta} \delta_\lambda (\kappa^* + \Phi^* - \nu D^*) + \hat{P}_\lambda \right], \quad (22)$$

$$\Delta v = -\Delta t \left[\mathcal{X}(u^*, \Delta t; \eta^\theta) + \frac{1}{R_e \Delta \theta} \delta_\theta (\kappa^* + \Phi^* - \nu D^*) - \hat{P}_\lambda \right], \quad (23)$$

where $\mathcal{X}(\eta^\theta)$ and $\mathcal{Y}(\eta^\lambda)$ are time-accumulated fluxes of η^θ and η^λ as per Eqs. (20) and (21) using an intermediate velocity u^* and v^* ; D^* is an approximation for the divergence at the cell corners, and $\hat{P}_\lambda, \hat{P}_\theta$ are the cell mean zonal and meridional pressure gradient terms. In addition to this overall time step, there is a $\Delta t/2$ predictor step to determine u^* , v^* and κ^* on the grid with an offset of $\Delta \lambda/2$ and $\Delta \theta/2$. Thus, this method is referred to as a *two-grid* and *two-time-step* method.

4 Data Dependency Analysis

The neighborhood of points needed for one iteration of Eqs. (18) and (19) is determined by the spatial accuracy order of the algorithm, Δt , and the geographical separation of the grid points, as dictated by the dimensionless Courant numbers:

$$C^\lambda = \frac{u \Delta t}{R_e \Delta \lambda \cos \theta}, \quad C^\theta = \frac{v \Delta t}{R_e \Delta \theta}. \quad (24)$$

In latitude θ , the geographical separation is constant. Therefore, if Δt is chosen appropriately, and the wind speeds, u and v , remain in an atmospherically realistic range, only the accuracy order of the transport algorithm is significant. Thus there are limited north-south neighbor dependencies (1, 2, or 3 lines of latitude) on each level.

A similar statement for the horizontal transport calculation in λ is, on the other hand, not possible. In order to accommodate the ‘pole problem’ of converging meridians near the pole, a quasi-Lagrangian approach [9] is employed.¹ This approach varies slightly from the classical semi-Lagrangian formulation, which determines where a departure point arrives after being advected for a time Δt . Instead, this formulation determines how the mass in the departure cell is distributed after a time step Δt . This formulation is crucial because it ensures that mass is conserved, while the classical formulation does not.

For the quasi-Lagrangian approach in [9], any given longitudinal Courant number $C_{i-1/2}^\lambda$ at the edge between cells $(i-1)$ and (i) can be greater than one,

¹ This approach for longitudinal flow should not be confused with the floating Lagrangian coordinate discussed earlier, which is purely vertical and ensures that the problem decouples into a set of 2-D problems.

and thus written as,

$$C_{i-1/2}^\lambda = K_{i-1/2} + c_{i-1/2}, \quad (25)$$

where $K_{i-1/2}$ is the integer part of $C_{i-1/2}^\lambda$ and $c_{i-1/2} = \text{mod}(C_{i-1/2}^\lambda, K_{i-1/2})$ is the fractional Courant number. Similarly, the flux at the left edge of the cell is decomposed into two parts:

$$\mathcal{X}_{i-1/2} = (\text{integer flux})_{i-1/2} + (\text{fractional flux})_{i-1/2}. \quad (26)$$

While the details of this algorithm are not important here, the upshot is that the method has dependencies on grid points which are at geographical distances dictated by the departure cell for a given Δt . Near the poles, this set goes well beyond the immediate east-west neighbors. These data dependencies are illustrated in Fig. 1.

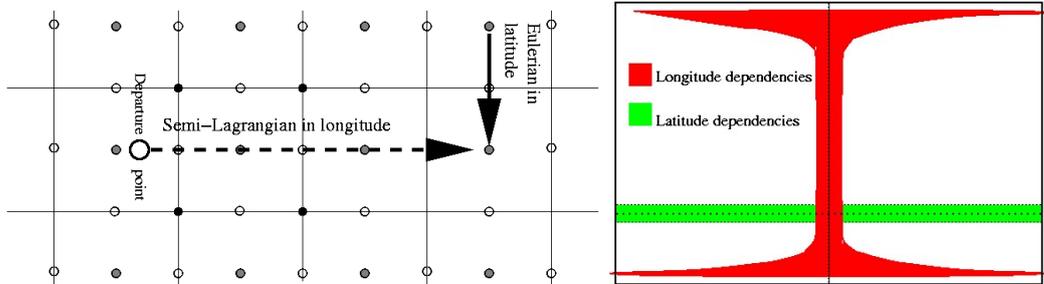


Fig. 1. The diagram at left depicts the different algorithms used for transport of vorticity near the poles in the finite-volume scheme. An Eulerian algorithm is used in the latitudinal direction, while a quasi-Lagrangian is used in the longitudinal direction. The departure cell in the quasi-Lagrangian algorithm can be many cells away from the target. This gives rise to data dependencies (right) which are fixed in latitude (i.e. nearest north-south neighbors independent of latitude, given here for $-60^\circ S$), while the west-east data dependencies (here for the prime meridian) are a function of latitude. This is only an approximate illustration; the precise data dependencies are also a function of the wind variables and will change between time steps.

As most atmospheric models, CAM is implemented on state-of-the-art distributed memory parallel supercomputers. The successful parallelization of the FV dynamical core is determined to a large extent by the required data traffic between processors. It is therefore imperative to locate the data on the processors in a way which minimizes inter-processor communication. Judged from this standpoint alone, the ideal data layout, or *domain decomposition*, would appear to be a set of latitude slabs, or *decomposition elements* (DEs), as illustrated in Fig. 2. Any given latitude resides in only one DE, avoiding the issues associated with the quasi-Lagrangian scheme for east-west transport. Furthermore, any given vertical profile is also in one DE, important for the

vertical integrations (such as the solution of the hydrostatic equation) and all of the physical parameterizations in CAM.

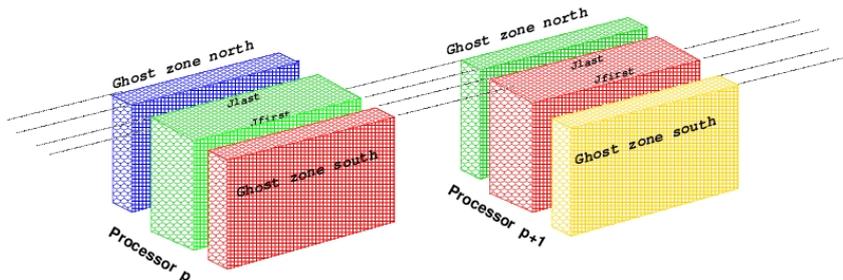


Fig. 2. *The 1-D algorithm decomposes the latitude-longitude-level domain into a set of latitude slabs. Each slab has a north and south halo region, which covers the latitudinal data dependencies. These halo regions are filled (‘halo exchange’) in a communication phase which can be overlapped with unrelated calculations. The calculation on haloed arrays can then take place subsequently without further communication.*

The 1-D decomposition was successfully used in CAM and other applications of the FV dynamical core. Unfortunately, with growing computer size, the inherent parallelism is no longer sufficient for very large multi-processor machines. For such machines it is necessary to decompose the data in a second dimension. But which dimension is ideal? Cutting in longitude creates excessive inter-processor dependencies near the pole, while distributing the vertical components creates extensive communication for vertical integrations and in the physical parameterizations.

We have chosen a technique with two alternate domain decompositions. The domain is decomposed by latitude and level (*lat-lev*) to calculate updates (22) and (23) in the Lagrangian frame. For the remainder of the dynamical core and in all of the physical parameterizations, the domain is decomposed in latitude and longitude (*lat-lon*). The use of two domain decompositions requires the constituent arrays to be redistributed or *transposed* from *lat-lev* to *lat-lon* space before the vertical calculations and back thereafter. Transposed-based methods, although expensive in terms of communication, have proven to be scalable [5] in the parallelization of a spectral model. A *lat-lev* decomposition has been illustrated as viable [15] in another spectral model. We have implemented the proposed technique, putting much emphasis on a highly optimized transpose operation.

5 Parallel Implementation

The 2-D domain decomposition is extensively discussed in [11]. Only the key points will be treated here.

The FV dynamical core consists of a component (referred to hereafter as `cd_core`) which operates on L independent levels in the vertically Lagrangian frame to essentially solve Eqns. (10), (11), (13), (14), and (15), and a remapping algorithm (hereafter `te_map`) which consists of the vertical remapping [7] from the Lagrangian frame back to the original vertical coordinate. It is also necessary to solve the hydrostatic equation (`geopk`) at each step, which is inherently a vertical integration. Finally, tracers are advected horizontally in `trac2d`. All of these components were first parallelized with the OpenMP shared memory multitasking paradigm [4], and obtained respectable performance on up to 16-32 CPUs on an SGI Origin 2000 [14].

The initial FV dynamical core implementation utilized two-sided communication — the only type supported by the MPI-1 standard [12] — namely `MPI_Isend` and `MPI_Irecv` primitives. Furthermore it utilized both a send buffer into which the data to be transferred are packed, and a receive buffer, from which the data are unpacked. An optimization to this defined MPI derived datatypes for the send and receive descriptors, circumventing the user buffers.

The code was then upgraded to use the enhanced MPI-2 standard which offers one-sided communication through the `MPI_Put` and `MPI_Get` primitives. One-sided communication requires MPI-2 *windows* to define the segments of memory that receive remote data. These windows can utilize the MPI derived datatypes described previously. Several communication approaches using MPI-2 were implemented, among them a straightforward scheme (*Method A*) utilizing both a send and receive buffer, and an optimized scheme (*Method B*), needing only a receive buffer. Other methods utilizing both MPI-2 and derived datatypes were also implemented. Unfortunately, this combination of functionality is not currently supported on the SGI Origin 3800 target platform.

In addition, the possibility is available on some platforms to multithread the one-sided communication. This can be done in different ways. First, a large block to be sent from a given DE to another is broken into a set of smaller blocks; the delivery of this set with `MPI_Put` is multithreaded. The second possibility is to multithread the delivery of all blocks from one DE with `MPI_Put` over the set of DE targets.

The `geopk` calculation is a vertical integration within the dynamics calculation taking place at a point where the 2-D domain decomposition is *lat-lev*. The original approach was to transpose the necessary arrays before and after this operation. As an additional optimization, a parallel algorithm which constructs and sends partial sums ‘upward’ was developed. This method does not require a transpose — it only requires the communication of the partial sum to all ‘higher’ subdomains. The partial sum method gives round-off differences

in the results with different DE configurations, due to the varying order of additions. A quadruple precision mode for debugging² purposes is available to ensure bit-wise reproducibility over all possible parallel configurations.

6 Results

The FV dynamical core was tested at both $0.5^\circ \times 0.625^\circ$ and $1^\circ \times 1.25^\circ$ horizontal resolutions, containing 576×361 and 288×181 grid points, respectively. 26 vertical levels were used for both resolutions. The target platforms were the SGI Origin 3800 *Chapman* (at NASA GSFC) with 1024 CPUs @ 600 MHz, and an IBM SP *Seaborg* (at DOE NERSC) with 380 Nighthawk nodes, each with 16 CPUs @ 375 MHz.

The 1-D decomposition was extensively evaluated with various numbers of latitude slabs and OpenMP threads per slab, using different communication primitives. Tab. 1 gives an overview of the timing results for the entire FV dynamical core in CAM for MPI-1 with intermediate buffers, MPI-1 with derived datatypes, and MPI-2 method A. The benefits of MPI-2 multithreaded communication are alluded to already in this comparison. Closer investigation of the communication timings indicates excellent speedup in the halo exchange. These results are in line with those found in [13].

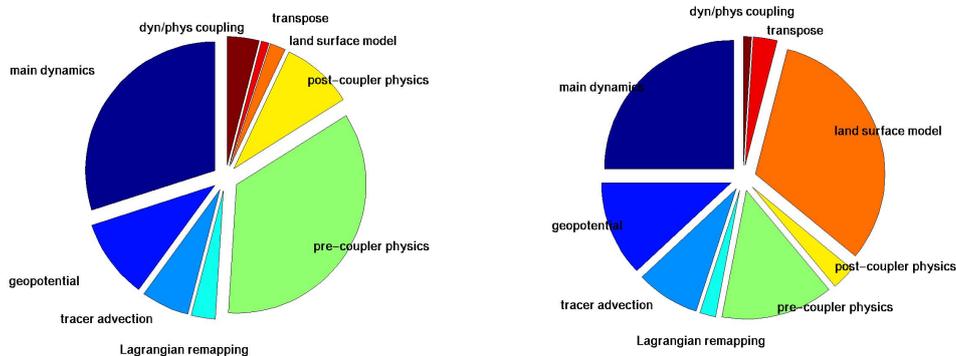


Fig. 3. The performance of the overall CAM application on both a 32 and 2944 CPU configuration (IBM SP ‘Seaborg’) is broken down by components. The main dynamics component (`cd_core` without `geopk`), the geopotential calculation (`geopk`), tracer advection (`trac2d`), and Lagrangian remapping (`te_map`) all scale better than average. The land-surface model has the poorest scaling due to insufficient computational load; the other pre/post-coupler physics scale better than average, in part thanks to their communication-free nature. With the targeted optimizations, the transpose and dyn/phys coupling do not present a performance bottleneck.

² Quadruple precision implies the communication of twice as much data, and is therefore not used for optimized benchmarking and production runs.

Table 1

The FV dynamical core (1-D decomposition) timings are given for a one-day CAM simulation at $1^\circ \times 1.25^\circ \times 26L$, run on configurations with 9, 18, 36, 45 DEs, each running with 1, 4 or 9 OpenMP threads, on an SGI Origin 3800 ('Chapman'). The MPI-1 methods using send and receive intermediate buffers or derived datatypes, are compared with MPI-2 Method A. The results indicate overheads for using MPI-2 one-sided communication with 1 thread, but better scalability for 4 and 9 threads than MPI-1. The OpenMP multithread performance of individual computation-only components is not affected by the communication paradigm. The increase in MPI-2 performance is thus attributable to the multithreading in the halo exchange communication.

DEs / Threads	MPI-1		MPI-2
	Buffers (s.)	Types (s.)	Method A (s.)
9 / 1	626	545	641
/ 4	193	194	187
/ 9	105	111	98
18 / 1	316	312	300
/ 4	112	111	102
/ 9	77	79	62
36 / 1	159	162	165
4	82	84	66
9	64	67	42
45 / 1	153	142	171
4	75	74	62
9	63	68	40

Fig. 3 compares the timing percentiles of various components of CAM in which the FV dynamical core is embedded. The figure indicates that the components scaling the worst and best are part of the physical parameterizations, which are outside of the dynamical core. Most physical parameterizations scale well because they are communication-free. However, some parameterizations, like short-wave radiation, are known to be load imbalanced, making the scaling worse than one might expect. Load balancing mechanisms [16] have been built into CAM which partially counteract this problem. The land-surface model scales by far the worst of all components and is a known bottleneck at very large processor count. Although it is not communication-bound, the land-surface model has insufficient computational load per land point in that regime and consequently consumes a much larger fraction of the overall run time. All

components of the dynamical core scale reasonably to 2944 CPUs, including the transpose, which consists entirely of communication. The worst performer is the geopotential calculation, while the best is the `cd_core` routine (without `geopk`).

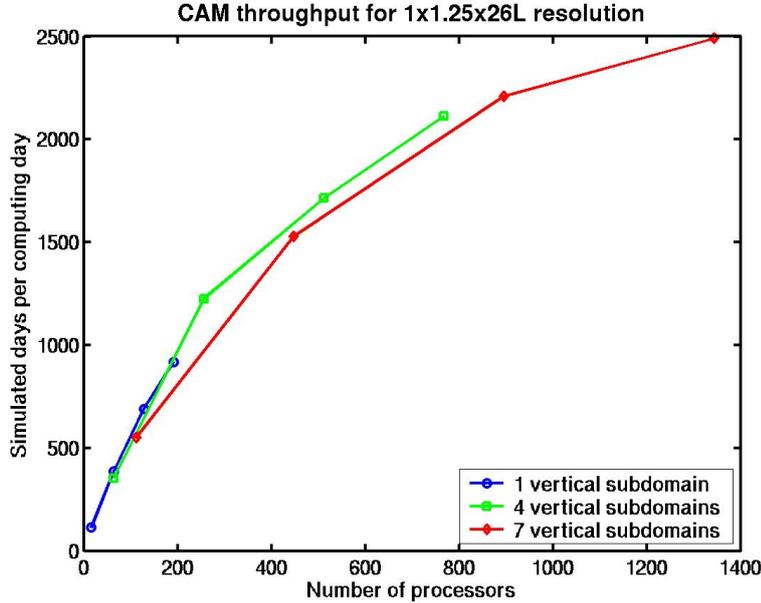


Fig. 4. The results on the IBM SP with Nighthawk 16-way nodes: the $1^\circ \times 1.25^\circ \times 26L$ resolution illustrates the scalability of the hybrid-parallel approach on a very large machine. Even though the resolution of this simulation can be considered low, the 2-D domain decomposition with 1 thread per DE (leftmost curve) allows parallelism to be exploited up to 200 CPUs, with 4 subdomains in Z (center curve) up to 780 CPUs, and with 7 subdomains (longest curve) up to 1320 CPUs.

Fig. 4 illustrates the overall scalability of the CAM run, in simulated days per day of wall-clock time. This includes all components illustrated in Fig. 3. Even for the relatively low resolution of $1^\circ \times 1.25^\circ \times 26L$, the 2-D hybrid-parallel implementation can exploit parallelism up to a large extent of the machine.

The MPI-2 multithreading capabilities of the code can also provide improved performance if these facilities are supported by the target platform. Fig. 5 illustrates a non-negligible performance increase for the overall FV dynamical core. The performance gains for the transpose (Tab. 2) were more modest than those for the halo exchange, but showed a marked improvement of method B over both method A and the MPI-1 default. The fact that one-sided communication is of less benefit to the transpose calculation is under investigation.

The partial sum optimization of `geopk` mentioned in Sec. 5 also achieved a notable performance improvement. As indicated in Tab. 3, the partial sum method (with roundoff error) performs consistently as good or better than the transpose approach.

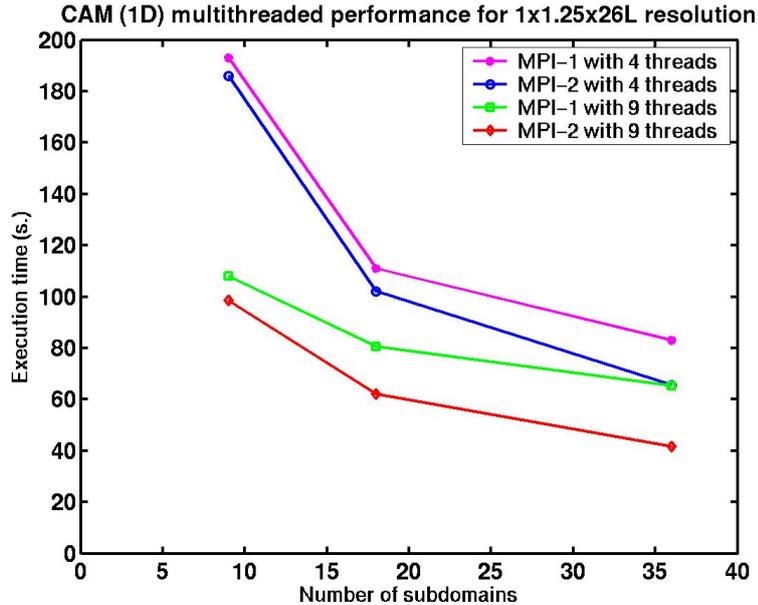


Fig. 5. SGI Origin 3800 results: the wall-clock time for the overall FV dynamical core (1-day simulation at $1^\circ \times 1.25^\circ \times 26L$ resolution) is given as a function of the number of subdomains (DEs) using a 1-D decomposition. For 4 and 9 threads per DE (upper and lower pair of curves, respectively), the MPI-1 (upper curve in pair) and MPI-2 (lower curve in pair) performances are given. MPI-2, which can take advantage of the multithreading in the communications primitives, can yield as much as a 20% overall reduction in computation time for 4 threads, and a 33% reduction for 9 threads.

Table 2

Timings in seconds are given for the overall transpose times in a 1-day $0.5^\circ \times 0.625^\circ \times 26L$ simulation on the SGI Origin 3800 with 4 vertical subdomains and N_{lat} bundles of latitudes (i.e., # DEs = 4 x N_{lat}). MPI-2 multithreading can lead to higher performance than the best MPI-1 method: MPI-2 method A is comparable to MPI-1 (using derived datatypes). Method B consistently outperforms both.

N_{lat}	MPI-1	MPI-2	
	Types	Method A	Method B
9	113	117	99.5
18	68.8	69.5	60.8
36	46.4	47.4	42.4

7 Conclusions and Future Work

We have presented a scalable parallel implementation of a finite-volume solver of the primitive equations. This has been fully integrated into the Community Atmosphere Model, and is available for use in the research community.

Table 3

Timings in seconds for the geopotential calculation in `geopk`: the partial sum method is as good or better than the transpose method, particularly if multiple threads per DE are employed.

	Threads pe DE		
	1	4	7
Transpose	59.7	51.2	36.6
Partial sum	60.0	30.1	30.3

Some additional optimizations to this implementation are ongoing. We are porting the code to the Cray X1, utilizing both vector parallelism and the SHMEM library for communication. Our primitives for irregular communication are thus being extended to use SHMEM as an alternative to MPI-2.

Acknowledgments We would like to thank Shian-Jiann Lin and William Putman for allowing us to use the OpenMP-parallel FV dynamical core and the `mod_comm` library as a basis for our development. We are also indebted to Jürg Schmidli and Heinz Blatter for their careful proofreading of this manuscript.

This work was performed under the auspices of the U.S. Department of Energy by the University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48, and NASA’s Goddard Space Flight Center, NASA Task #00-9103-01a.

This is LLNL Report UCRL-CONF-214013.

References

- [1] A. Arakawa and V. Lamb. A Potential Enstrophy and Energy Conserving Scheme for the Shallow Water Equations. *Mon. Wea. Rev.*, 109:18–36, 1981.
- [2] M. B. Blackmon, B. Boville, F. Bryan, R. Dickinson, P. Gent, J. Kiehl, R. Moritz, D. Randall, J. Shukla, S. Solomon, G. Bonan, S. Doney, I. Fung, J. Hack, E. Hunke, and J. Hurrell. The Community Climate System Model. *BAMS*, 82(11):2357–2376, 2001.
- [3] P. Colella and P. Woodward. The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations. *J. Comp. Phys.*, 54:174–201, 1984.
- [4] L. Dagum and R. Menon. OpenMP: An Industry-Standard API for Shared-Memory Programming. *IEEE Transactions on Computational Science and Engineering*, 5(1), 1998.

- [5] J. Drake, I. Foster, J. Michalakes, B. Toonen, and P. Worley. Design and Performance of a Scalable Parallel Community Climate Model. *Parallel Computing*, 21(10):1571–1592, 1995.
- [6] E. Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge University Press, London, 2003.
- [7] S.-J. Lin. A 'Vertically Lagrangian' Finite-Volume Dynamical Core for Global Models. *Mon. Wea. Rev.*, 132(10):2293–2307, Oct. 2004.
- [8] S.-J. Lin. A finite-volume integration method for computing pressure gradient force in general vertical coordinates. *Q. J. R. Met. Soc.*, 123:1749–1762, 1997.
- [9] S.-J. Lin and R. B. Rood. Multidimensional Flux-Form Semi-Lagrangian Transport Schemes. *Mon. Wea. Rev.*, 124(9):2046–2069, Sep. 1996.
- [10] S.-J. Lin and R. B. Rood. An explicit flux-form semi-Lagrangian shallow-water model on the sphere. *Q. J. R. Met. Soc.*, 123:2477–2498, 1997.
- [11] A. A. Mirin and W. Sawyer. A Scalable Implementation of a Finite-Volume Dynamical Core in the Community Atmosphere Model. *Int. J. for High Perf. Comp. Appl.*, 19(3), Aug. 2005.
- [12] MPI Forum. MPI: A message-passing interface standard. Technical Report UT-CS-94-230, Univ. of Tennessee, 1994.
- [13] W. M. Putman, S.-J. Lin, and B. Shen. Cross-Platform Performance of a Portable Communication Module and the NASA Finite Volume General Circulation Model. Submitted to *Parallel Computing*, 2004.
- [14] W. Sawyer. A Multi-level Parallel Implementation of the Lin-Rood Dynamical Core. Presentation at the *8th Workshop on the Solution of Partial Differential Equations on the Sphere*, 1999.
- [15] J. G. Sela. Weather Forecasting on Parallel Architectures. *Parallel Computing*, 21(10):1639–1654, 1995.
- [16] P. H. Worley and J. B. Drake. Performance Portability in the Physical Parameterizations of the Community Atmospheric Model. *Int. J. for High Perf. Comp. Appl.*, 19(3):1–15, August 2005.