Journal Title
XX(X):2–30

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Performance analysis of fully explicit and fully-implicit solvers within a spectral-element shallow-water atmosphere model

K. J. Evans [1], R. K. Archibald[1], D. J. Gardner[2], M. R. Norman[1], M. A. Taylor[3], C. S. Woodward[2], and P. H. Worley[1]

Abstract

Explicit Runge-Kutta methods and implicit multistep methods utilizing a Newton-Krylov nonlinear solver are evaluated for a range of configurations of the shallow-water dynamical core of the spectral-element Community Atmosphere Model to evaluate their computational performance. These configurations are designed to explore the attributes of each method under different but relevant model usage scenarios including varied spectral order within an element, static regional refinement, and scaling to the largest problem sizes. This analysis is performed within the shallow-water dynamical core option of a full climate model code base to enable a wealth of simulations for study, with the aim of informing solver development within the more complete hydrostatic dynamical core used for climate research. The limitations and benefits to using explicit versus implicit methods, with different parameters and settings, are discussed in light of the trade-offs with MPI communication and memory and their inherent efficiency bottlenecks. Given the performance behavior across the configurations analyzed here, the recommendation for future work using the implicit solvers is conditional based on scale separation and the stiffness of the problem. For the regionally refined configurations, the implicit method has about the same efficiency as the explicit method, without considering efficiency gains from a preconditioner. The potential for improvement using a preconditioner is greatest for higher spectral order configurations, where more work is shifted to the linear solver. Initial simulations with OpenACC directives to utilize a GPU when performing function evaluations show improvements locally, and that overall gains are possible with adjustments to data exchanges.

Keywords
global climate modeling, implicit methods, Newton-Krylov, regional refinement, GPU acceleration

Introduction

Motivated by investigations of the multiscale behavior of the global atmospheric system, global and regionally refined atmosphere climate models are being developed to produce climate length simulations on leadership computing facilities at sub 40 km horizontal spacing (Small et al. 2014; Roberts et al. 2015; Jung et al. 2012). At this spatial resolution, some global and regionally-refined climate and weather models avoid time step size limitations due to the well-known CFL constraint by using some form of subcycled explicit time integration. In this case, chemical tracers, dynamics, and hyperviscosity subcomponents are subcycled with time steps that are a fraction of the size of the cloud physics parameterization scheme (Neale et al. 2010; Zarzycki et al. 2014). However, it has been shown that subcycling can present new errors and instabilities associated with time-splitting in related applications (Estep et al. 2008). Other models use a form of semi-implicit methods (Sandbach et al. 2015; Åström et al. 2012), which take advantage of scale separation to isolate and solve the relatively fast linear gravity waves efficiently. For high resolution configurations, semi-implicit schemes still require subcycling within subcomponents, and that can result in reduced solution accuracy, depending on the splitting strategy (Knoll et al. 2003). Nonetheless, these time integration strategies within atmosphere models have so far enabled high-fidelity simulations that effectively use multi-core petascale computing systems (Dennis et al. 2012).

The computing environment beyond petascale is markedly different because power constraints are driving supercomputer hardware to transition to more power efficient GPU and manycore systems. This difference creates more complexity, but also opportunity to explore and revisit algorithms that might utilize these new systems effectively. The Community Atmosphere Model (CAM), the atmosphere component within the Community Earth System Model (CESM) and the Accelerated Climate Model for Energy (ACME), uses a Runge-Kutta (RK) based explicit method for time integration. As mentioned, these methods require a time step restriction for stability that poses a strong barrier to throughput. Furthermore, the next generation ACME model and other atmosphere components within coupled models are being developed with more vertical levels, additional cloud physics complexity within a vertical column, and the option to use localized spatial refinement to capture local climate changes within

[1]Oak Ridge National Laboratory [3]Sandia National Laboratories [2]Lawrence Livermore National Laboratory

Corresponding author:
Katherine J. Evans, Climate Change Science Institute, Oak Ridge National Laboratory, 1 Bethel Valley Rd. Mailstop 6301, Oak Ridge, TN, USA.
Email: evanskj@ornl.gov

the global flow. Therefore, they will have a different performance profile going forward. A solver that leverages computing power differently has the potential to be useful with these future models. To investigate, we explore the computational performance of explicit and implicit methods by applying them to a suite of configurations.

CAM shallow-water dynamical core

We utilize the shallow-water approximation to Navíer Stokes flow to analyze solver performance because many of the complexities of the model in which we are interested, including spatial and temporal discretization, library interfaces, and regional refinement, can be explored without the cost of multiple vertical layers. Of course, moving to the three-dimensional hydrostatic model will change the performance profile. Where relevant, predicted performance changes with extension to the hydrostatic model are discussed below.

A very general expression for the partial differential equations of fluid dynamics solved within CAM, referred to as the dynamical core, or 'dycore,' can be expressed as,

$$\frac{\partial \mathbf{x}}{\partial t} = \mathbf{G}(\mathbf{x}), \tag{1}$$

where $\mathbf{G}$ is the sum of the nonlinear terms operating on the state vector, $\mathbf{x}$. For shallow-water, the horizontal velocity components, $\mathbf{v} = (u, v)^T$ in the x and y coordinates along each cube face direction, explained later, and the thickness of the fluid z, where $H = z + z_s$ is the total height of the fluid and z_s is the bottom topography, comprise the state vector, $\mathbf{x} = (\mathbf{v}, z)^T = (u, v, z)^T$. The shallow-water operator on $\mathbf{x}$ as defined within (1) is given by,

$$\mathbf{G} = \begin{pmatrix} -(\zeta + f_{cor})\hat{k} \times \mathbf{v} - \nabla\left(\frac{1}{2}\mathbf{v} \cdot \mathbf{v} + gH\right) \\ -\nabla \cdot (z\mathbf{v}) \end{pmatrix}. \tag{2}$$

The terms that make up $\mathbf{G}$ are presented here in continuous form because the focus is on time discretization. The first term on the right-hand-side contains the vertical component of the three-dimensional relative vorticity,

$$\zeta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}, \tag{3}$$

and the Coriolis acceleration, f_{cor}. In (2), $\hat{k}$ is the unit vector normal to the surface of the sphere, g is gravitational acceleration, and ∇ refers to the horizontal gradient.

CAM parametrizes energy diffusion from the resolved scales of motion to close the fluid system with a combination of terms to achieve a favorable scale selection. The goal is to minimize energy dissipation and provide a minimal sponge layer near the model top. Referred to as 'hyperviscosity,' the diffusion terms within the shallow-water and hydrostatic dycores in CAM are applied as part of the full discretized solution to (1). Both the degree of damping and the level of scale selection are determined by using a resolution-dependent diffusion coefficient ν with a fourth order gradient (second order Laplacian) as detailed in Jablonowski and Williamson (2013). By

default, shallow water CAM uses an explicit Runge Kutta time stepping algorithm with the hyperviscosity terms applied using Forward-Euler as needed to maintain stability, using the same value of ν for both the velocity and thickness terms. We did not apply hyperviscosity within the implicit solution method. The only test cases where the lack of hyperviscosity is an issue are the regionally-refined cases, and this is discussed with the results below.

Spatial discretization

The spatial accuracy and distribution of grid points influences the accuracy and ease of solution for a given time stepping method, and so we present a brief description of the spatial discretizations used here. The default two-dimensional, horizontal discretization scheme in CAM consists of a cubed-sphere grid mapped to the sphere as in Rancic et al. (1996). The cube faces are subdivided into fully unstructured but quasi-uniform, quadrilateral elements that can be varied to alter the spatial resolution either globally or locally as in Guba et al. (2014). Within each element, points on the cube face are further defined using a continuous spectral finite-element Galerkin scheme. This version of CAM is referred to as 'CAM-SE.' The nodal two-dimensional space within each element is spanned by piecewise Lagrange interpolating polynomials applied to a tensored grid of Gauss-Legendre-Lobatto (GLL) point values.

Additional details regarding the specifics of the spectral element discretization within CAM and its general attributes for global atmospheric modeling are presented in Taylor et al. (2007, 2008); Taylor and Fournier (2010).

For the test cases presented here, the spatial domain is discretized with polynomials of degrees 3, 7, and/or 15 within each element (which provide 4th, 8th, and 16th order accuracy, respectively). These are illustrated by each square in the bottom portion of Figure 1, where the left/blue, middle/green, and right/red represent the approximate distribution of points within an element, respectively. Note that the higher-order polynomial options exhibit a more severe clustering of points, which is associated with reduced stability for CFL limited time-stepping methods and increased computational intensity within an element, regardless of the choice of time-stepping scheme. The overall horizontal spatial resolution of the model can be altered by either changing the number of elements, the polynomial order, or both, and these choices impact the computational speed and spatial convergence characteristics. A more complete discussion of the trade-offs for a CFL-limited method is given in Taylor et al. (1997).

The nomenclature to refer to the spatial resolution of a CAM-SE configuration is as follows: a uniform resolution around the sphere is named for the element count along the edge of a single cube face, 'ne,' and the order of accuracy of the polynomials within each element, 'np.' For example, a resolution of CAM-SE that is commonly used for global climate studies and that closely matches the 1° finite volume grid spacing (Evans et al. 2013; Zarzycki et al. 2014) is referred to as 'ne30np4.' It has 30 elements along the edge of a cube face (ne30) and 3 independent polynomial points per element (4th order, np4). Around the equator there are 4 cube faces, so the total number of points is $4 \times 30 \times 3 = 360$, which is 1° spacing. The total number of elements around the globe for this example is $30 \times 30 \times 6$, or 5,400. Whereas, a case labeled 'ne15np8' has half the elements along the edge of a cube face but a similar, albeit slightly more

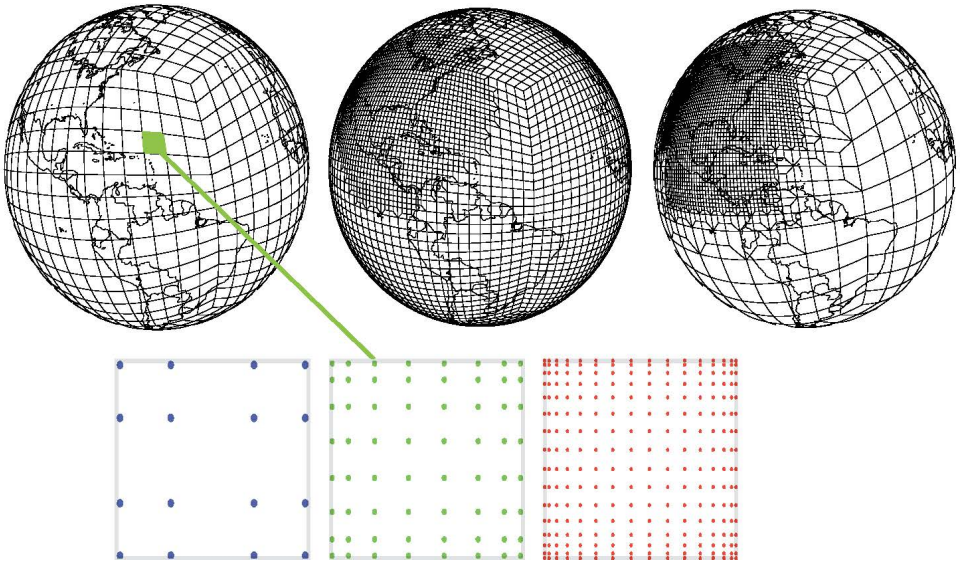

Figure 1. Illustration of the mesh for the uniform (left, ne15 shown for clarity), single refinement (middle), and multiple refinement (right) options for element discretization used for the MTN, MTN$\times 2$, and MTN$\times 8$ cases (top row). An illustration of the Gauss-Legendre-Lobatto nodes within each element are shown on the lower row for 4^{th} (left/blue), 8^{th} (middle/green), and 16^{th} (right/red) order. Varying orders and element layouts of the CAM-SE grid are used to create 'equivalent' resolutions as in Figure 3.

fine-scaled resolution. There are 7 independent polynomial points per element, and therefore somewhat more points around the equator, $4 \times 15 \times 7 = 420$. The number of elements could be further reduced to achieve the closest equivalent resolution to the ne30np4, although for ease of comparison this study altered the configurations by factors of two. For regionally refined element configurations, the resolution is named for the nominal global value of 'ne' plus the multiplier of refinement. There are two configurations with regional refinement analyzed in this study, one with a single level of refinement going from ne30 to ne60, named 'ne30$\times$2np4,' and one with 8 levels of refinement going from ne10 to ne80, named 'ne10$\times$8np4.'

Time discretization options

Within the shallow-water dycore of CAM-SE, there are multiple time integration options implemented from which to compare. These include the traditional leapfrog (LF) method long used and understood within large scale atmospheric models, an explicit Runge-Kutta (RK) scheme, and two fully implicit options. For context, results using the fully-explicit, centered LF method

$$\frac{\mathbf{x}^{n+1} - \mathbf{x}^{n-1}}{2\Delta t} = \mathbf{G}(\mathbf{x}^n). \tag{4}$$

along with the Robert filter (Robert 1966),

$$\bar{\mathbf{x}^n} = \mathbf{x}^n + \frac{1}{2}\nu(\mathbf{x}^{n+1} + \mathbf{x}^{n-1} - 2\mathbf{x}^n), \tag{5}$$

are presented for some cases. The filter is used at each time step to prevent oscillatory growth of an unstable mode at the $2\Delta\mathbf{x}$ level. Using this filter for any number of steps reduces the LF method accuracy to first-order.

The default time integration method in the shallow-water dycore of CAM-SE is a second-order five-stage RK method, as presented in Dennis et al. (2012). For concise use in the code, this method is implemented in CAM-SE such that the first two stages are given by,

$$\frac{\mathbf{x}^{n_1} - \mathbf{x}^n}{\frac{1}{5}\Delta t} = \left(\frac{1}{2}\right)\mathbf{G}(\mathbf{x}^n) \tag{6}$$

$$\frac{\mathbf{x}^{n_2} - \mathbf{x}^n}{\frac{1}{5}\Delta t} = \mathbf{G}(\mathbf{x}^{n_1}) \tag{7}$$

where the subscript on the time level, n, indicates the stage, and $\mathbf{x}^{n_2} = \mathbf{x}^{n+1}$. The next three stages are equivalent to the LF method, as in equation 4, but using the stage times step size, $\frac{1}{5}\Delta t$.

As is well-known already, LF is more efficient and therefore interesting to include in a performance analysis, however the RK method allows for reduced subcycling and tracer-mass consistency within the hydrostatic version of CAM (Dennis et al. 2012). Hyperviscosity is applied after each time step for the LF method and after 4 of the 5 stages of the RK method.

The Backward-Euler, BDF2 (BD) and Crank-Nicholson (CN) fully-implicit time discretization methods are also available within CAM-SE. Both BD and CN are analyzed for this study and are given by

$$\frac{\mathbf{x}^{n+1}}{\Delta t} - \left(\frac{4}{3}\right)\frac{\mathbf{x}^n}{\Delta t} + \left(\frac{1}{3}\right)\frac{\mathbf{x}^{n-1}}{\Delta t} = \left(\frac{2}{3}\right)\mathbf{G}(\mathbf{x}^{n+1}) \tag{8}$$

for BD, and

$$\frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t} = \left(\frac{1}{2}\right)\mathbf{G}(\mathbf{x}^{n+1}) + \left(\frac{1}{2}\right)\mathbf{G}(\mathbf{x}^n), \tag{9}$$

for CN. Although both are second order and A-stable, they have different characteristics that affect convergence and performance. As implemented in CAM-SE, CN recomputes $\mathbf{G}(\mathbf{x}^t)$ for each residual calculation within an iteration rather than saving it because the residual call is controlled by a solver library. Whereas with BD, only the current time level is needed for $\mathbf{G}(\mathbf{x})$. Although BD requires the storage of three time levels of $\mathbf{x}$, these levels are retained for all methods in CAM-SE because the size of the state vector is relatively small.

Solution methodology for the implicit scheme

For both BD and CN, an inexact Newton-Krylov method is used for solution of the algebraic, nonlinear systems at each time step. For BD, the nonlinear residual is

$$\mathbf{F}(\mathbf{x}^{n+1}) = \frac{\mathbf{x}^{n+1}}{\Delta t} - \left(\frac{4}{3}\right)\frac{\mathbf{x}^n}{\Delta t} + \left(\frac{1}{3}\right)\frac{\mathbf{x}^{n-1}}{\Delta t} - \left(\frac{2}{3}\right)\mathbf{G}(\mathbf{x}^{n+1}) \tag{10}$$

and for CN,

$$\mathbf{F}(\mathbf{x}^{n+1}) = \frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t} - \left(\frac{1}{2}\right)\mathbf{G}(\mathbf{x}^{n+1}) - \left(\frac{1}{2}\right)\mathbf{G}(\mathbf{x}^n). \tag{11}$$

The components of $\mathbf{F}$ with the spectral element derivatives are provided in Section 3 of Lott et al. (2015). The mechanics of Newton's method is explained in detail elsewhere (Knoll and Keyes 2004), but the key parameter that optimizes the balance between efficiency and providing consistent and robust convergence is the nonlinear convergence tolerance, η_{nl}. In the tests that follow, the Newton iteration is terminated when the L_2 norm of the nonlinear residual drops by a factor of $\eta_{nl} = 1.0 \times 10^{-3}$ below its initial value.

Within each nonlinear iteration, we use the GMRES Krylov method (Saad and Schultz 1986) to solve the linear approximation. GMRES finds a solution to the linear system within a space spanned by a Krylov subspace. Within each iteration, a basis for that subspace is increased by one vector, and that vector is orthonormalized against the prior basis vectors. GMRES only uses matrix information through a matrix-vector product, and since this matrix is the Jacobian of the nonlinear residual, we approximate this action by a finite-difference with the residual. The choice of differencing parameter can have a modest effect on performance as discussed in (Woodward et al. 2015). However, we use the default value in Trilinos for this work. Since the algorithm increases the subspace by one basis vector each iteration, memory constraints are an issue. While restarting the algorithm can reduce the memory requirements, reduction of iterations is key. The two main drivers for algorithm cost within the linear solver are the matrix vector multiply (function evaluations) and the orthogonalization (MPI reduce). Details about this sensitivity are discussed within the results.

The degree to which the update of the state vector should be approximated is given by

$$\|\mathbf{J}(\mathbf{x}^k)\delta\mathbf{x}^k + \mathbf{F}(\mathbf{x}^k)\|_2 < \eta_k \|\mathbf{F}(\mathbf{x}^k)\|_2, \tag{12}$$

where $\mathbf{J}$ is the Jacobian of nonlinear residual $\mathbf{F}$, $\delta\mathbf{x}$ is the iteration update, and η_k is the linear tolerance criteria. The process to attain convergence of (12) drives the efficiency, memory use, and robustness of the method. For the cases presented below, we found efficient and still robust convergence when we set the linear convergence tolerance to $\eta_k = 1.0 \times 10^{-4}$. For stiffer configurations, the maximum iteration count is reached before linear convergence, but this did not prevent overall nonlinear convergence. It is not reasonable to test all the implicit solver parameters for each test case to find the most efficient.

Details about the implicit solver implementation in CAM-SE have been presented elsewhere (Evans et al. 2009). In summary, these methods use C++ interface modules and the 2003 Fortran bind(C) attribute to connect to the C++ based Trilinos library (Heroux et al. 2005) of solvers, specifically using NOX and Belos (Bavier et al. 2012) packages for the nonlinear and linear solvers, respectively. These packages, in turn, call back to CAM-SE for the Fortran-based routines. Logistically, the solver libraries receive settings to define the solver and its parameters from an xml input file, for example to access the nonlinear tolerance criteria, η_{nl}, for convergence. Our choices for each test case are mentioned along with the results in the *Performance analysis* section

and are selected to optimize performance and robustness. The current limitations of using a solver library are discussed in the *Discussion* section.

Performance analysis

With the goal of exploring a wide range of behavior for the time integration methods presented above, multiple tests with a diversity of configurations across them are analyzed. All the simulations presented here have been executed on the Oak Ridge Computing Facility's Titan supercomputer. Titan is a hybrid CPU-GPU system with 16 cores per node and sufficient nodes to execute the large-scale problems we present below, including those with GPU acceleration. We experienced ongoing system variability in the total solution times as high as 10%. Most of the variability was experienced between runs submitted as different job submissions for larger jobs. To minimize variability, all but the most expensive runs for a given simulation type were submitted together for all methods compared. Therefore, the absolute solution times for a given configuration for all methods should be considered approximate, but the relative costs for a given set of simulations across the methods compared did not vary significantly. For this study, we do not present the effects of changing parameter values of the methods when it did not affect performance outside the range of variability.

For several test cases commonly used within the climate model development community to assess new methods, it is not appropriate to compare performance with implicit methods. For example, test case '2' from Williamson et al. (1992), is steady state and thus can be solved by BD or CN within 1 time step. Test case '1' from the same analysis also does not mimic typical climate configurations used with the full hydrostatic model because it is a simplified case of pure advection.

The test cases selected for this analysis exercise the full nonlinearity of the shallow-water equations and have already demonstrated accuracy for larger time step sizes (Evans et al. 2010; Jia et al. 2013). The first, named "MTN" herein, simulates zonal flow over the sphere with a single mountain feature creating fluid displacements, as presented in Williamson et al. (1992) as test case '5'. We show results with a uniform resolution to isolate effects of a wide choice of time integration methods within a typical global climate model configuration. Results are also presented for MTN with two mesh-refined configurations demonstrating how methods are influenced by variably-sized unstructured meshes. The other test case examined here was developed and presented by Galewsky et al. (2004), named "GSP" herein, and has been designed to explore multiple scales of barotropic flow instabilities. This test matches similar instability test cases used to examine the hydrostatic dycore, so it provides insight into expected behavior for implicit methods within that flow regime. With this test case, we examine solver scaling performance for a variety of grid configurations utilizing traditional and alternative spatial layouts. In addition, we present and analyze performance of the MTN test case using GPU accelerators via an implementation of OpenACC directives. This matches concurrent efforts of Norman et al. (2015) in the tracer-advection kernel of CAM-SE. This work extends initial efforts to examine acceleration of this same test case using GPUs with CUDA (Archibald et al. 2015). The motivation to transition to OpenACC from CUDA is to enable more code portability and prepare for possible use of OpenMP4.

Significant effort was expended to maximize efficiency for each solver, with one exception. In earlier work, a preconditioner was developed for the shallow-water model; it strongly reduced iteration count and scaled well with problem size Lott et al. (2015). Efforts to make this preconditioner cost effective per call are possible (e.g. Yang et al. (2010)), but substantial, and so its development and optimization proceeds only within the hydrostatic dycore, the eventual target for performance improvement. Given that preconditioners reduce solver expense by reducing linear iterations, a solver that is comparable in cost without a preconditioner, or at a minimum, shifts the majority of work to the linear iterations, shows maximum potential for success within the hydrostatic CAM-SE.

The L_2 norms of temporal error are calculated and presented with the performance results below (except MTN, which is presented in Evans et al. (2010)). The time step convergence rate for each method evaluated here is also verified (not shown), using the RK method for the reference solution. It is well known in the model development community that the cloud physics as currently implemented in CAM-SE is not time step converged (e.g. Williamson (2013); Mishra and Sahany (2011)), but since we are restricting this study to the dynamical core, the results presented here are time step converged. The dynamic properties and nonlinearity of the problem drive robustness and govern the magnitude of the temporal error relative to error from other parts of the model such as spatial resolution and/or diffusion and limiters. In practice, one should run the model with the largest time step size that produces a level of error below other sources of error in the simulation, as presented in Vandewalle et al. (1991) and Jakob et al. (1995). Low order time step convergence and error levels close to 1.0×10^{-2} for the stratiform cloud physics in CAM-SE (Wan et al. 2015) allow the dynamical core setup to use a rather coarse solver tolerance of 1.0×10^{-3} in the dynamics and maintain solution accuracy. Note that the convergence tolerance is not the same as the time discretization error. The tolerance determines whether the nonlinear residual norm evaluated with the current approximate solution shows sufficient decrease. The nonlinear residual is a discretized form of the mathematical model. The time discretization error measures how well the temporal discretization scheme estimates the actual, or continuous solution. Given that one role of the MTN case is to explore conservation properties, we verified that the mass and energy residuals are not impacted by the choice of solver and were consistent with the previously reported values for the CAM-SE model after 15 days (Evans et al. 2010).

For all test cases, the LF and RK methods used the largest evenly divisible time step that is stable for the complete test length. For the implicit CN and BD schemes, either a 30-minute (1,800 seconds(s)) or 20 minute (1,200 seconds(s)) time step is used, in order to match the time step generally used by the cloud physics and chemistry components. These time step sizes resolve the diurnal cycle and are the maximum time step size across all the components in CAM-SE.

Uniform MTN case

This 15 day MTN case is configured with a 1° ne30np4 spatial discretization, and Figure 2 shows the solution for the fluid thickness with the same projection as the grids shown in Figure 1, for comparison. Table 1 shows the solution time for simulations

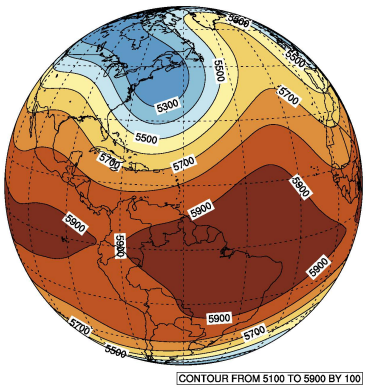

Figure 2. Fluid thickness (m) after 15 days of simulation for the MTN case using the BD method. All methods produce the same visual result.

using the LF, RK, CN, and BD schemes as outlined in the *Time discretization* section. The largest stable time step size for the LF and RK methods is 120 s for this case. All the run time costs are rounded to the nearest number of seconds. All of the MTN simulations used 1 CPU node with 16 cores per node. Only the LF scheme includes hyperviscosity. In general, there is a reduction in stability and smoothness by not including the hyperviscosity terms, however this is not an issue for the MTN case except using LF. The time to solution is broken down in Table 1. The overall time is given by *Total Run Time* and *Solver* encompasses the time spent providing the solution at the next time level. Within the solver, the time spent on *Fcn Evaluations* is presented in terms of number count (#) and time spent (s) executing all the residuals calls as in (10) and (11). *Orthog* is the time spent orthogonalizing each new Krylov vector added within the GMRES solver.

Time step (s)	120	120	120	1800	1800	1800	1800
Compiler	GNU	GNU	PGI	GNU	GNU	PGI	PGI
Method	LF	RK	RK	CN	BD	CN	BD
Total Run Time (s)	74	123	149	488	295	575	353
Solver call (s)	70	119	147	473	280	547	314
Fcn Evaluations (s)	24	118	146	192	112	247	137
Fcn Evaluations (#)	10,801	54,000	54,000	75,984	47,426	75,984	47,422
Linear its/ts (#)	-	-	-	97.4	58.8	97.4	58.8
Orthog (#)	-	-	-	188	106	205	163

Table 1. CAM-SE performance for 15 days of the MTN case with 1° resolution, 5,400 elements, and an np=4 configuration for a range of time stepping options, using one node and 16 CPU cores. s=seconds, '*GNU*' and '*PGI*' refer to the compiler types, and 'Method' is the time integration scheme, with more descriptions provided the text. The time step size refers to the full Δt step.

Comparing explicit and implicit schemes for MTN in Table 1 shows material differences in cost. The code is 67% less efficient with RK than LF, and this is easily explained by the 88% more function evaluations performed by RK over the course of the simulation and the corresponding 85% more time within that part of the code. As for the relative performance of the two implicit schemes, BD is only 58% of the cost of CN even though both are second order. Given that BD requires 62% fewer function evaluations than CN to converge, the reduced iteration count explains almost all of the cost differences. The extra cost to CN to evaluate the spatial terms at the previous time level, as in (9), is not a major factor in performance. Some numerical analysis of the BD and CN methods exists regarding relative stability and performance, e.g. Dharmaraja (2007); Vandewalle et al. (1991). The results presented above are consistent with their work in that the stability region with CN may not be as well defined near the edge due to undamped oscillations in the solution (Dharmaraja 2007). That said, the cases are different enough that a comparison is minimally useful. We are not aware of any published results that present and/or analyze these differences in terms of iteration count and performance for PDE problems of this size and complexity.

The larger differences among the algorithms presented for the uniform MTN solution are between the implicit and explicit schemes. The BD scheme is about 2.4 times more expensive than the RK scheme for this configuration and processor count. Because BD is able to take a much larger time step than RK, it is able to achieve a solution with slightly fewer total function evaluations, even though it must take about 2 nonlinear iterations and 58 linear iterations per time step. The function evaluations within the nonlinear and linear solver comprise only 40% of the total solver cost for BD, so the extra costs are contained in the other parts of the Newton solve. The linear solver cost for this simulation is 257 s, so it is 92% of the total solver cost of 280 s. It is optimal to shift the majority of work from the nonlinear to linear solver. Within the linear solver, the two main steps are the matrix vector multiply and the orthogonalization, and they cost 140 s and 106 s, or about 55% and 41% respectively, of the 257 s total for this case. Within the multiply, most of the cost is due to the function evaluations, taking 112 s of the 140 s for this simulation. Note that because this case is parallel only within one node, all the MPI communication is done within DRAM and thus is minimal. The MPI reduce within the orthogonalization step is about 4% of the total cost. For the MTN case, the iterated modified Gram-Schmidt method (IMGS) orthogonalization within the Trilinos Belos GMRES linear solver option is the most efficient. IMGS is similar to the classic Gram-Schmitt method (ICGS), but with modifications that provide stability. It is about 25% cheaper than both ICGS and the default option, DGKS, which is ICGS with a DGKS correction step (Daniel et al. 1976). Several different block size settings, which determine the number of iterations before restarting, did not significantly affect the efficiency of this problem.

Compiler and Programming Environments

Another choice that materially affects performance on Titan is the compiler. Table 1 shows that simulations executed with the GNU/4.9.0, or '*GNU*' compiler as compared to the PGI/15.7.0, or Portland Group, '*PGI*' compiler are consistently more than 10% faster for all the time integration choices, and sometimes, as with the MTN

case using BD as shown in Table 1, over 25% faster. The degree of difference depends on simulation parameters including time step size, length of simulation, discretization, etc. Some care was taken to use compiler flags that would optimize performance, particularly with *PGI* because it is slower. It is difficult for an application scientist to know if a build has been optimized for performance, especially when using supercomputing systems with compiler wrappers that themselves change build time options. The *PGI* compiler also takes significantly longer to build than *GNU*, although the degree to which they vary is dependent on build options. Because CAM-SE is a Fortran based code, *PGI* is an important choice on Titan even though it is slower. The accelerator programming options for Fortran (CUDA and OpenACC) are not yet available for *GNU*. The implicit solver uses a Fortran interface to the C++ based Trilinos third-party library, so it is possible that differences in the compiler ability to optimize C++ as well as Fortran are affecting the build and run performance.

Differences due to the choice of compiler for this case are not as great as the choice of which implicit time stepping method is used, or implicit versus explicit, but they are substantial and robust for many processor layouts, compiler settings, and extend to other cases presented below. The cost differences based on compiler choice are distributed through various parts of the simulation. Note also that the choice of compiler impacts the number of iterations the solver takes in some cases. For the MTN case using the BD method, the simulation using the *PGI* compiled code needed slightly fewer function evaluations than the *GNU* compiled code, as shown in Table 1. These differences originate from the PGI double versus GNU quadruple level of precision of the GLL quadrature calculation, which in turn produces different nodal values within each element at round-off levels. Therefore, the initial residual and problem to be solved is altered. These results highlight the sensitivity of the solver to small changes in values within certain parts of the code. The interpretation of array types could be synced for multiple compilers, but it is important to mention this sensitivity to the compiler and simulation parameters here for awareness by large scale coding projects using multiple compilers and iterative solvers within legacy code.

MTN Case with refined grids

The MTN case is also configured with a locally refined mesh around the area of the mountain to better resolve the flow features in that vicinity. The nature of the mesh refinement within CAM-SE and its application in the shallow-water dynamical core are explained in detail in Guba et al. (2014). The grids are generated using the cubit mesh generation algorithm to create the transition region between the two grids. The cubit method creates slightly more distorted elements than the more recently developed SquadGen algorithm (refer to Figure 5 from Guba et al. (2014)), however the more challenging case provides a relevant comparison for various time stepping algorithms. We explore the behavior of time stepping methods applied to two refinements. The first, "MTN$\times$2," represents a global 1° uniform resolution (ne30np4) with a local refinement to a uniform 0.5° resolution (ne60np4) in the area around the mountain feature. This case has 6,418 elements, which is a slight increase from the 5,400 elements in the ne30np4 uniform case presented above. However, it is has many fewer points than the 21,600 elements that would be needed to achieve the same 0.5° resolution with

a uniform grid. The second refined case, "MTN×8," refines the grid by a factor of 8 ranging from approximately 3° (ne10np4) to 0.375° (ne80np4) and allows us to investigate a high connectivity case. Because it is locally refined from a coarser grid than the uniform MTN example, it has only 2,990 elements versus 5,400. The problem size is about half that of the uniform 1° case, but the points are substantially redistributed to provide greater detail in one localized area.

It has been demonstrated that with implicit methods, hyperviscosity or other similar diffusion/filters are not needed with the same frequency or intensity as with the explicit schemes for cases with evenly distributed elements (Evans et al. 2010; Jia et al. 2013). However, results of the MTN×2 and MTN×8, and early work with the hydrostatic core using the BD method, indicate the need for some form of diffusion for spatial smoothness for long time integrations. With the assumption that the hyperviscosity operator will not be needed frequently for the stable and nondispersive BD schemes, it is activated for the MTN×2 and the MTN×8 for only the explicit methods. Also, these cases are run for fewer days to avoid issues with noise for any of the schemes.

Table 2 shows the relative performance of all the methods for the MTN×2 case. The simulation time is given after 5 days, and this case is run using 2 nodes and 16 CPU processors per node (32 total). Similar performance differences due to compiler choice are present for this case as with MTN. Because the hyperviscosity operator is called twice for each RK solve compared to once after each LF time step, the RK method is relatively slower by a factor of 2.8 compared to LF for the MTN×2 case. Table 2 presents the costs for the explicit method to perform the hyperviscosity calculations.

The spatial grid refinement of MTN×2 and ×8 increases the problem stiffness, so maintaining the same large time step size for the implicit methods as with MTN comes at the cost of extra iterations. A maximum linear iteration count of 30 is set for BD and CN was found to give the best efficiency and stability, rather than requiring a tolerance criteria. In this case, the linear solution update used by the nonlinear solver is less exact and shifts the work from from linear to nonlinear iterations. Implicit solvers can be altered significantly to change the iteration statistics, and the most efficient results here are the best we found after fairly comprehensive testing. Both BD and CN took 6

Time step (s)	60	80	1800	1800
Method	LF	RK	CN	BD
L2 Norm	1.8×10^{-5}	1.8×10^{-5}	3.9×10^{-4}	4.4×10^{-4}
Total Run Time (s)	24	68	147	108
Solver call (s)	20	63	137	101
Fcn Evaluations (s)	6	23	41	32
Fcn Evaluations (#)	7,201	27,000	32,240	23,888
Hyperviscosity (s)	13	40	-	-

Table 2. Performance results for 5 days of the MTN case with 1 level of mesh refinement (MTN×2) from 1° (ne30np4) to 0.5° (ne60np4) executed on 2 nodes each with 16 CPU cores, all using the *GNU* compiler. The L2 norm and convergence stats are given after 1 day of simulation. The BD and CN schemes as configured converged in 6 nonlinear iterations. Other parameters, including layout, are the same as set for simulations in Table 1.

nonlinear iterations per time step to reach convergence for these settings. Comparing the implicit BD to the explicit RK method, the expense ratio is reduced to 1.58 with MTN$\times$2, however the cost reduction is largely associated with use of hyperviscosity for the RK scheme. If we assume a similar need for hyperviscosity in the BD scheme as with LF (once per time step), then the cost ratio of BD to RK rises to 1.8, which is still a lower relative cost than in the MTN case. The change in problem size and using multiple nodes also affects the relative performance, but to a lesser degree.

Also like the MTN case, we found that CN is slower than BD due to additional function evaluations within the linear solver. The BD method is 74% of the cost of CN for MTN$\times$2, and as with the MTN case, BD takes 88% as many iterations as CN does. The fact that the the BD method is relatively closer in cost to CN compared to MTN is associated with the relative decrease in linear iterations for this particular configuration. The convergence tolerances can be adjusted to shift work to the linear solver, but doing so reduces efficiency for this setup. Like the MTN case, the IMGS orthogonalization option provided the best efficiency (20% faster than the default DGKS) for the MTN$\times$2 case.

Many of the results from the $\times 2$ refined case are similar for the $\times 8$ refined case, shown in Table 3, even though there are now 8 levels of refinement. The BD and CN implicit methods are able to converge to a similar solution, with the same time step size and solver parameter settings as with MTN$\times$2. Although the problem is more stiff, the problem size reduction results in a faster solution for all methods, and a relatively similar efficiency for the RK and BD schemes. Unlike the MTN and MTN$\times$2 cases, the ICGS option within Belos provided better performance for the stiffer MTN$\times 8$ problem. This is consistent with the results in Frayssé et al. (1998), who found that ICGS performed better when the problem is not well preconditioned or does not scale well. Ultimately, the best choice of orthogonalization will depend on the preconditioner, so this choice is noted as a starting point for future studies within the hydrostatic core. For the strongly regionally refined configuration (MTN$\times 8$) investigated here, the implicit BD and CN methods are closer in performance to the RK method, to the point that the

Time step (s)	30	40	1800	1800
Method	LF	RK	CN	BD
L2 Norm	6.4×10^{-5}	2.9×10^{-6}	4.0×10^{-4}	4.0×10^{-4}
Total Run Time (s)	19	54	89	57
Solver call (s)	15	50	85	52
Fcn Evaluations (s)	5	18	24	14
Fcn Evaluations	14,401	54,000	42,928	27,216
Hyperviscosity (s)	11	32	-	-

Table 3. Performance results for 5 days of the MTN case with multiple levels of mesh refinement (MTN$\times 8$) from 3° (ne10np4) to 0.375° (ne80np4), executed on 2 nodes each with 16 CPU cores, all using the *GNU* compiler. The L2 norm and convergence stats are given after 1 day of simulation. The BD and CN schemes as configured converged in 6 nonlinear iterations. Other parameters, including layout, are the same as set for simulations in Table 2.

differences are within the same range of performance as other sources of variability, e.g. compiler choice and parameter settings. The L2 Norm of the solution for the implicit schemes after day 1 of the simulation, as presented in tables 2 and 3, are acceptable relative to other sources of error, but they will need to be revisited for climate length runs, going forward.

Barotropic instability case (GSP)

The next test case we analyze was developed to mimic the complex and multiscale flow instabilities that develop within global atmospheric flow over 6 days. An initial perturbation is instigated within a balanced, but unstable barotropic, or horizontally variable, flow field. Within several hours of simulation time, the model exhibits gravity wave excitation. Then, over a period of 6 days, vorticity anomalies develop that are akin to those within realistic atmosphere models. A full description, with initial values and perturbations, is given in Galewsky et al. (2004), and it is referred to here as the GSP case. Accurate solutions to this test case using implicit methods have demonstrated their ability to step over time scales of the initial, much faster gravity wave generation while still resolving an accurate longer term roll up of the vorticity field (Jia et al. 2013). Here, we adjusted the problem size from ne24np8 to ne30np8 to enable easy comparison covering a range of resolutions commonly used by the climate model community.

The fine scale structure of the flow in this test case presents an opportunity to explore the performance impact by altering the spectral discretization within each element of CAM-SE, which is represented by 'np.' The top three panels of Figure 3 show the vorticity field at day 6 for three GSP simulations with an approximate 0.5° resolution with low (ne60np4), medium (ne30np8), and very high spectral order (ne15np16), all using the BD implicit time integration method with a time step of 1,200 s. The three spectral order options and how they map to each element within the cubed sphere are shown for reference in the bottom three squares in Figure 1. To maintain a similar nominal resolution, the number of elements along the edge of a cube face are are reduced by a factor of 2 when the spectral order is doubled. Due to shared points at the cube edges, the degrees of freedom are more than doubled when the spectral order is doubled. Note that the fine scale structure and noise within the vortical structures matches solutions produced with an equivalent resolution of a fully spectral model used to illustrate the test case in Figure 4 of Galewsky et al. (2004) as well as previous analyses of the accuracy of time stepping methods in Jia et al. (2013). The bottom 2 panels of Figure 3 show the differences between the simulations shown in the top three panels, ne15np16 - ne60np4 (4th panel from the top) and ne30np8 - ne60np4 (bottom panel), using the same contour levels. It is clear that the spatial configuration associated with the different layouts of 'ne' and 'np' do not produce a materially different solution.

Table 4 displays run times for the BD and RK methods for the medium spectral order layout, 'ne30np8,' using 30 nodes and 16 CPU processors per node (480 cores). To evaluate the methods using more MPI parallelism, we assigned this test case more nodes. BD is slower by a factor of 1.75 (*GNU*) and 1.5 (*PGI*) compared to RK for this setup. Note the relative increase in cost for *GNU* over *PGI* compared to the MTN and refined MTN test cases for the overall runtime. The cost to initialize the quadrature,

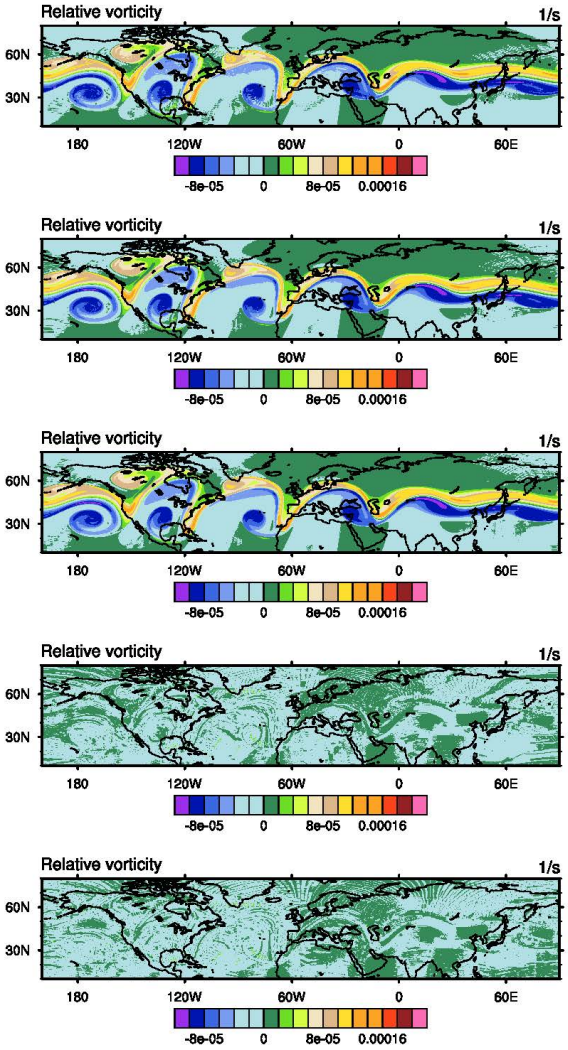

Figure 3. Vorticity field (s^{-1}) of the horizontal flow field at 6 days for GSP simulations with 3 different layouts of the spectral element grid, ne60np4 (top), ne30np8 (2nd from top), and ne15np16 (middle), that all have a nominal resolution of 0.5°. The bottom 2 panels show differences between the simulations shown in the top three panels, ne15np16 - ne60np4 (4th panel from the top) and ne30np8 - ne60np4 (bottom panel), using the same contour levels. Note the fine scale structure and noisiness of the fluid flow; these are discussed in the text.

"gvinit," is more expensive for *GNU* with higher spectral discretization because it interprets the functions at quad-precision as coded currently in CAM-SE. This extra

cost up front is amortized over long runs and so it is minimal for production runs. Within the main time stepping portion of the model, cost differences between *GNU* and *PGI* are also reduced as compared to the MTN cases.

Because the GSP configuration presented in Table 4 uses a higher spectral order, the problem is more stiff and the time step size differential between BD and RK is larger. As a result, there are fewer function evaluations for BD than RK. On the other hand, MPI communication costs within the orthogonalization step limit the throughput of the BD method when using more cores, as shown by the parallel reduction costs within the linear solver, refer to 'MPI Allreduce' in Table 4. Given the results from the MTN$\times 8$ case and some smaller test runs of GSP (not shown), we use the ICGS option for all GSP configurations. As with the mesh refined cases, we do not use hyperviscosity with the BD simulations, including those shown in Figure 3. Hyperviscosity is responsible for about two-thirds of the cost of the RK method, and so the ability to call it only occasionally is an important consideration for using BD versus the RK method going forward.

A closer look at the cost differences for all the GSP configurations provides some insight into the cost to simulate the GSP case with higher order and finer spatial resolutions. The MTN and GSP cases are both configured with 5,400 elements, but GSP has about 6 times the degrees of freedom within an element, $(np-1)^2$ (see Figure 1). Noting the clustering of edge points with higher order, the RK method must take a time step about 1/4 as large as with the MTN case for the GSP simulations to remain stable. For BD (and CN, not shown), this clustering requires about a 5 times increase in linear iterations for convergence. The iteration count and MPI communication, rather than the time step size, limit weak scaling in the case of implicit methods. This is discussed more in the next section.

Time step (s)	20	20	1800	1800
Method	RK	RK	BD	BD
Compiler	GNU	PGI	GNU	PGI
L2 Norm	1.3×10^{-7}	1.3×10^{-7}	2.5×10^{-4}	2.5×10^{-4}
Total Run Time (s)	144	159	255	244
Solver call (s)	127	157	239	242
Fcn Evaluations (s)	32	39	58	51
Fcn Evaluations (#)	2.6×10^5	2.6×10^5	2.1×10^5	2.1×10^5
MPI Allreduce (s)	-	-	72	76
Orthog (s)	-	-	145	144
Gvinit (s)	16.0	1.1	16.0	1.4
Hyperviscosity (s)	92.0	110.0	-	-

Table 4. Performance results for GSP after 6 days using RK and BD using the *GNU* and *PGI* compilers for an ne30np8 configuration, using 30 nodes and 480 CPU cores.

Alternative Options for Scalability

The performance of GSP test case configurations with various settings of element count and spectral order are analyzed as a way to better understand expected model performance for larger scales. CAM-SE is typically configured with an np = 4 spectral decomposition for high resolution simulations to maximize the element-based MPI parallelism and time step size on CPU-dominated computing systems (Dennis et al. 2012). Presently, the model has also been configured with fewer elements and a corresponding increase in the spectral order to provide an equivalent resolution but with higher spatial order for comparison, as shown in Figure 1. Without the severe time step limitation experienced with explicit LF and RK time stepping algorithms, we explore the potential that higher spectral order may provide better accuracy and/or performance on some computing platforms, such as those where the increased intensity can favorably target cost via thread-level or hybrid parallelism.

The solution times for 6 day simulations of the GSP test case are presented in Figure 4 for a suite of simulations covering a range of elements and using three spectral order decompositions, np = 4, 8, and 16. The spatial resolutions range from 2 to approximately 1/8° spatial resolution, depending on the layout. Table 7 provides the time step size for each when using the RK method. Simulations with the same spectral order are connected as lines with matching colors/shapes for the RK (solid) and BD (dashed) time stepping methods. Every simulation is assigned 6 elements per core and is run with 16 cores per node. For example, configurations with 384 elements (ne8) are assigned 8 nodes with 128 cores, and those with 245,600 elements (ne120) are assigned 1,800 nodes with 28,800 cores, etc. Each line represents the weak scaling behavior of CAM-SE for both time stepping methods for a given spectral order.

Unlike the previous test cases, the degree to which the runs using the BD method in Figure 4 could be optimized for performance is more limited because the larger runs are quite expensive. For robustness and consistency across all simulations, a smaller time step of 1,200 s and the classic Gram-Schmitt (ICGS) orthogonalization is used. In addition to showing better behavior for stiff problems, ICGS has shown better performance at larger scales compared to other orthogonalization choices, because it gathers dot products to ensure more computational work and reduced communication (Frayssé et al. 1998). Consistency across simulations provides the most insight for the BD method performance behavior and its trends for this analysis, even if efficiency is not optimized for each run. That said, the optimization efforts used to maximize performance for the smaller cases presented above are leveraged here and used where possible across all the runs. An 1,800 s time step size converged and is faster for most of the configurations, but the ne30np16 case requires the 1,200 s time step to converge. The GMRES restart length, named ”block size” within Belos, is set to 20, and the linear tolerance is set to a constant value of 1.0×10^{-3}, although in some cases the runs reached the maximum iteration count.

The explicit RK and unpreconditioned implicit BD methods are subject to the same increasing stiffness of the problem with increasing resolution and do not weak scale. Computation time increases as the number of elements are increased (moving right along a given line) even though the same number of cores are assigned, regardless of problem size. Computation time also increases as the spectral order is increased

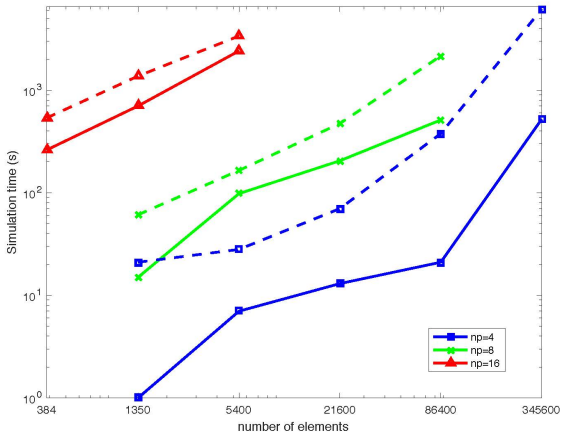

Figure 4. GSP test case for 6 days of simulation over a range of problem sizes (# elements) for the np=4, 8, and 16 spectral decompositions using the RK (solid) and BD (dashed) methods. The nodes assigned for each simulation are provided in the text.

(moving up lines along the vertical axis). However, this is expected because the same number of cores are assigned to simulations with the same element count. That said, the ratio of expense of BD over RK decreases with increasing spectral order. The ratio of BD/RK drops from 4.0 for np=4, to 1.7 for np=8, and 1.5 for np=16 for the 'ne30' (5,400 total elements) set of configurations.

The same configurations used to create Figure 4 can be grouped by 'equivalent resolution,' so that the difference in cost among simulations with similar problem sizes but varied element count and spectral order can be compared. Again, doubling the spectral order more than doubles the number of points due to the shared element edges, so these should be considered a significant upper limit of cost in terms of resolution equivalence. Figure 5 displays computation times of the RK (solid lines) and BD (dashed lines) methods for the same suite of configurations presented in Figure 4, but with the same number of nodes/cores assigned to configurations with the equivalent resolutions of 1, 0.5, and 0.25°s. These are 8/128, 32/512, and 128/2,048 for the 1, 0.5, and 0.25° configurations respectively. The x-axis marks the number of elements along an edge of a cube face for a given simulation, so moving right along the x-axis corresponds to a decrease in spectral order to maintain the same approximate resolution, as in Figure 1.

Although equivalently sized problems used the same number of processors, the higher spectral order configurations (with corresponding lower element count) take longer to complete. For RK and BD, this increase in time is due to the increased number of time steps and linear iterations, respectively. Table 5 provides the cost for the RK simulations when normalized by the number of time steps per hour. For the 1 and 0.5° runs, the np=16 runs are almost twice as expensive as np=8 and 4. However for the 0.25° configuration, the cost between np=16 and np=4 is about the same, and cheaper per time step for np=8. For BD, the equivalent comparison is to

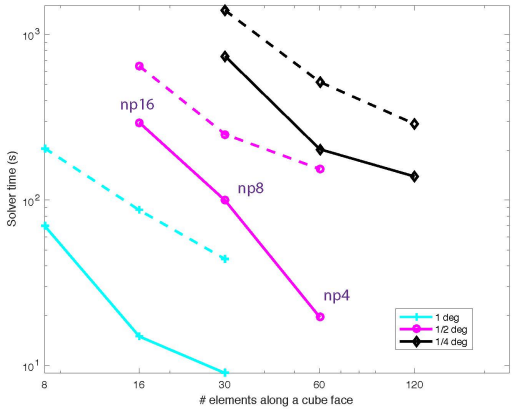

Figure 5. GSP test case after 6 days of simulation using the RK (solid) and BD (dashed) methods with 'equivalent' resolutions for 1, 0.5, and 0.25°, grouped by color/shape. The number of nodes used is constant for each, as listed in the text. The x-axis marks the number of elements along a cube face used for a configuration plotted with a given line, and each line has configurations with np=4, 8, and 16. For the 0.5° RK cases, the np of each simulation is provided next to the results to show how the equivalent simulations vary in np.

normalize by the number of Krylov iterations per time step, as presented in Table 6. The normalized cost is similar per Krylov iteration, regardless of spatial layout, for the 0.5 and 025° simulations, with np=8 being the most efficient. The cost is about equal for np=8 and np=4 for the 1° simulations. The impact on performance due to increasing spectral order is experienced by both the RK and BD methods, but the growth in computational intensity plays only a minor role as the spatial layout is altered. In fact, the minimal cost of np=8 indicates a possible sweet spot of performance for high resolution configurations using the BD method.

A preconditioner has recently been implemented for the shallow-water test cases to determine scalability and potential for efficiency for the hydrostatic version of CAM (Lott et al. 2015). It is based on an approximate block factorization of a Picard linearization of the shallow-water equations and uses a common semi-implicit form of the equations. It has not been optimized for performance for shallow-water, but we can measure the growth in iteration count with the equivalent resolution runs to determine

Spectral order (np)	16	8	4
1°	9.3	5	4.5
0.5°	14.75	8.3	5
0.25°	12.35	6.8	11.6

Table 5. Cost (in seconds) for RK simulations of the GSP test case, with varying spectral order for the 1, 0.5, and 0.25 equivalent resolution configurations. These costs have been normalized by the number of time steps per hour.

if using this preconditioner will enable higher spectral order configurations to provide a reasonable alternative for running high spatial resolution simulations on non-MPI dominated machines. When the preconditioner is activated for the 1° simulations (lowest left dotted line in Figure 5), the average number of Krylov iterations per time step for the ne8np16, ne16np8, and ne30np4 configurations is reduced from 302.1, 168.9 and 102.9 to 6.2, 4.1, and 4.3 respectively. The function evaluation calls are reduced by a factor of 19.6, 13.0, and 7.6. The implications of this reduction are presented in the *Discussion* section.

GPU acceleration

The performance behavior of the solvers with different grid layouts is also of interest as we look toward computing architectures with manycore and/or GPU accelerators. The capability to use GPUs to accelerate the implicit solution method by transferring the majority of the residual calculation to the GPU has shown potential; initial performance using CUDA Fortran showed improved relative performance in the residual calculation as the number of elements per node or the spectral order within an element are increased (Archibald et al. 2015). Here, we extend this effort to show results using the MTN case with OpenACC directives rather than CUDA. OpenACC is potentially more portable (Sabne et al. 2014) and consistent with other ongoing OpenACC-based, GPU-enabled development in CAM-SE (Norman et al. 2015).

Similar to Table 1, Table 8 shows performance results for the MTN case, but with an ne30np8 layout. Also, an additional simulation using the BD method (far right) with OpenACC directives to execute the majority of the residual calculation, or function evaluation, on the GPU is included. Unlike the CPU-only simulations, the GPU simulation used 15 of the 16 cores per node to enable a core to interact with the GPU. Because *GNU* does not have the ability to use OpenACC right now, these simulations used the *PGI* compiler. The overall runtime using the GPU is substantially slower. However, the time spent only in calculating the residual on the GPU and not including time to transfer between CPU and GPU ("no trans"), is substantially faster than when using only the CPU. Because the function evaluation performs an MPI exchange of the element edge values at the end of the residual, and also because the residual values must then be passed through the Fortran data structures to the C++ based Trilinos library, solution data currently must be offloaded from the GPU to the CPU within each call of the residual. Related work to move other parts of CAM-SE to the GPU has shown that only MPI exchange information needs to be offloaded

Spectral order (np)	16	8	4
1°	0.74	0.59	0.57
0.5°	0.87	0.77	0.93
0.25°	0.73	0.66	0.89

Table 6. Cost (in seconds) for BD simulations of the GSP test case, with varying spectral order for the 1, 0.5, and 0.25 equivalent resolution configurations. These costs have been normalized by the number of Krylov iteration per time step.

# elem	384 (ne8)	1,536 (ne16)	5,400 (ne30)	21,600 (ne60)	86,400 (ne120)
np4	-	-	120	60	20
np8	-	80	20	8	-
np16	32	12	4	-	-

Table 7. Matrix of CAM-SE spatial configurations used to create the simulations displayed in Figures 4 and 5. Where provided, values indicate the time step sizes used by the RK method for each configuration. When using the BD method, all the configurations used a time step size of 1,200 s. An entry of '-' means the case is not run for that configuration. Configurations with similar resolutions are positioned diagonally from the bottom left entry to the top middle for 1°, and so on to the right for the 0.5 and 0.25°.

from the GPU (Norman et al. 2015). Using the configurations presented in Table 8, it is estimated that just offloading MPI exchange information as opposed to the entire solution state is about ninety times less expensive in terms of data transferred. This data transfer cost reduction will enable a competitive GPU implementation of the BD method within the hydrostatic CAM-SE. This work is informing the next steps to use the GPU and is spurring cooperative Trilinos/CAM-SE code development. Note that the kernel rewrites within the GPU enabled code have not been applied to the CPU only residuals, so the speed-up should be considered accordingly.

The portion of the residual that operates on the GPU strong scales with processor count for the smaller cases we ran (not shown). The data transfer portion does not scale, however, and renders the problem relatively slower with increasing processor count, compared to the CPU-only version. Implementing a data transfer of just the MPI exchange information should reduce this limitation, although not completely. Also, when the number of elements is increased using higher order configurations, these configurations reach memory limits of the GPU. This limitation is something to be investigated thoroughly before moving to the hydrostatic core.

Time step (s)	10	1800	1800
Method	RK	BD	BD GPU
Total Run Time (s)	116	196	811
Solver call (s)	115	191	805
Fcn Evaluations (s)	41	66	675
Fcn Evaluations, no trans (s)	-	45	17
Fcn Evaluations (#)	10,800	17,339	17,339

Table 8. Similar performance results for MTN using *PGI* as in Table 1 but with an ne30np8 layout, run for 1 day of simulation, and using the BD method. The simulations are completed using 2 nodes, and, for the case on the far right, the main portion of the function evaluation is calculated on the GPU. The timings for steps within the solver are discussed in the text.

Discussion

With a breakdown of the costs for a suite of simulations using the shallow-water dycore within CAM-SE, we show the source of performance contributions to inform further methods development at scale and to apply to the hydrostatic dycore. Optimization of each simulation has been attempted to the extent possible, although given the continual improvements within the solver library, compilers, etc., these results should be considered as snapshots along a constantly evolving development process. Where they exist, general trends that inform further development and deployment of implicit solvers for use on CPU and GPU systems are highlighted.

The higher order and regionally refined model configurations render a given test case more expensive to solve. For explicit LF and RK this expense is due to the time step restriction, whereas implicit BD and CN methods instead require additional iterations to reach convergence. The increase in FLOP count per degree of freedom due to increased np and the associated additional quadrature sweeps does not affect the runtime significantly because both CPUs and GPUs are good at doing FLOPs once the data is local. Plus, the vast majority of computational time in the dycore is spent in MPI data exchange for both CPU and GPU configurations. Given that the model errors within the physics portion of the hydrostatic model are larger than errors in the dynamical core, the lower temporal error associated with the time step size reduction for LF and RK is not appreciated. This opens the door for implicit methods to become competitive, especially at large scales, because the penalty for stiffness is smaller for implicit than explicit methods. This difference is illustrated above for strongly regionally refined and higher-order spectral test cases, whereby BD performance is not significantly more expensive than RK, even without a preconditioner. The hyperviscosity, which consumes 60% of RK runtime for the MTN$\times$8 case, is not included in the BD solve, and it will need to be included to an extent as yet to be determined. Also, the MTN$\times$2 and $\times$8 cases are only run on 2 nodes, so MPI latencies and bandwidth contentions are low. As this technology is transferred to the hydrostatic CAM-SE, there will be relatively much more data to transfer over MPI due to the contribution from multiple vertical layers.

For both implicit and explicit methods, the shift in grid points from elements to increased spectral order within an element increases the computational intensity. The resulting costs, using the same number of processors, are presented in Figure 5. As presented in the *Alternative options* section, the cost increases for the higher spectral order simulations for the same resolution. However, this increase is mostly due to extra time steps for RK and increased iterations for BD rather than additional computational intensity, at least going from np=4 to np=8. For this reason, more effective use of Titan for higher spectral order discretizations is indicated up to a point. If the iteration count of the BD method can be minimized, this could translate into more efficient solution times for CAM-SE.

Implicit methods contain a hierarchy of nonlinear and linear iterations, and the behavior and expense of each are defined by multiple parameters. For example, the choice of a loose linear solver tolerance improves efficiency, but can prevent overall nonlinear convergence. Similar efforts to tease out the performance trade-offs with iterative schemes also recommend the fewest iterations to maintain stability and yet

minimize cost (Sandbach et al. 2015). For the test cases analyzed presently, the optimal tolerance depends on the spatial grid and time step size. The method of orthogonalization for each linear iterate within GMRES also affects efficiency in the range of 20-30% and should be investigated more significantly in the future. We observed that ICGS outperformed the other orthogonalization methods for problems with significant regional refinement and higher-order spatial configurations. Because these iteration parameters depend on the model configuration and test case, it is not possible to optimize universally for each problem for implicit schemes. We expect the settings presented here can inform simulations within the hydrostatic CAM-SE dynamical core as well as similar partial differential equation based fluid flow solvers using these schemes.

For the regionally refined and GSP test cases, especially the MTN$\times$8 configuration, the BD method does relatively well in performance relative to RK. Addressing the degree to which a block-factored preconditioner could reduce the iteration count for these problems will determine the degree of expected improvement for the hydrostatic dynamical core. As presented in the *Scaling* section, a block preconditioner can reduce the iteration count by a factor of 49, 41, and 24, for 1 degree resolution configurations using np=16, 8, and 4, respectively. If the cost of forming the preconditioner is less than these factors, then the BD method is more efficient. Generally, we expect that the higher spectral order configurations will benefit more from a preconditioner because more of the cost is shifted to the linear solver.

As mentioned in the *GPU acceleration* section, the simulations using OpenACC and GPU acceleration within the residual calculation showed potential for gains, but with several major caveats. Faster performance could be gained with an improved or optimized *PGI* compiler or if the *GNU* compiler were to provide OpenACC capability. That said, the major gains in performance would be realized from minimizing the offloading of data from the GPU. There are efforts underway to enable the Trilinos solver calling the residual to host the state vector and other data on the GPU throughout the solve, using Kokkos (Edwards et al. 2012) and enabling the transfer of data across language interfaces while hosted on the GPU. The present results highlight the potential for performance benefits by using the GPU to accelerate the residual calculation and, eventually, other parts of the dynamical solver.

In addition to ongoing work to address interoperability of C++ and Fortran on the GPU, there is a need to address the very fragile and constantly changing build, link, and execute process when using the Trilinos package in CAM-SE, especially with OpenACC. Consistent versions of supporting libraries such as HDF5 usually must also be updated whenever changes to the compiler are made. Compiler support for OpenACC is new and still under development, so bugs are not infrequent. Selecting compiler flags that enable the code to build, execute, and provide optimized performance, is an ongoing challenge. As presented above, *GNU* is consistently faster than *PGI* with all time stepping schemes. Interactions with compiler vendors and computing centers provided material improvements to the code's robustness and efficiency presented above, however ongoing collaboration is needed to continue this trend as code, compiler, and library developments occur. One option to reduce complexity in the build system is to use a hand-crafted implicit solver rather than a solver library within the code. However, libraries prevent the need for significant

expertise in solver methods and their ongoing improvements and also reduce the code maintenance burden within the application code.

Conclusions

A robust comparison of time stepping methods applied to the solution of the shallow-water dynamical core in the CAM-SE model provides insight into the best use and optimization strategy for second order implicit time stepping methods for the eventual target, the hydrostatic dynamical core of CAM-SE. The BDF2 method is cheaper and takes fewer iterations than the Crank-Nicolson method for the range of problems we analyzed. Both implicit schemes become more competitive for the regionally refined MTN test cases, because an extreme time step reduction hinders the throughput of explicit methods. Exploring a range of spectral order configurations to identify more efficient configurations for a given spatial resolution, the higher spectral order np=16 and 8 showed some relative gains, although the increasing stiffness of the problem indicates the need for a preconditioner. We show evidence that a preconditioner could alleviate the efficiency bottleneck for implicit schemes, even if the performance gain is modest. Using the GPU within the nonlinear residual of the implicit algorithm shows speed-up within that portion of the code using OpenACC, however the cost of data exchanges needs to be addressed before the strategy should be implemented within the hydrostatic dynamical core. The lack of maturity in code building and executing for applications of this size and complexity, with multiple languages and libraries, is an ongoing issue to be addressed by large-scale computing facilities and the compiler vendor community.

Acknowledgements

This manuscript has been authored by UT-Battelle, LLC and used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, both of which are supported by the Office of Science of the U.S. Department of Energy (DOE) under Contract No.DE-AC05-00OR22725. This work was also partially supported under the auspices of DOE by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC, LLNL-JRNL-716595. Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy Office of Advanced Scientific Computing Research and Office of Biological and Environmental Research. The publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. We acknowledge a Hackathon event hosted by the Oak Ridge Computing Facility for the opportunity and assistance adding OpenACC directives to our code. We thank Oksana Guba for providing a script to create the refined grid plots and two reviewers for their helpful comments that improved the manuscript. The data from this paper can be reproduced using the branch_tags/2d_implicit tag from the HOMME model within the NCAR CESM repository, hosted at https://svn-ccsm-models.cgd.ucar.edu, once released by the CESM community.

Author Biographies

Katherine J. Evans is a group leader and senior research staff member for the Computational Earth Sciences Group at Oak Ridge National Laboratory. She received her Ph.D. from Georgia Institute of Technology and completed a postdoc at Los Alamos National Laboratory studying implicit numerical methods for phase change and fluid flows from 2004 to 2006. Kate's work includes high resolution and scalable numerical methods development for atmosphere models and verification and validation tools for global climate models. She is a member of the American Meteorological Society, American Geophysical Union, and the Society for Industrial and Applied Mathematics.

Patrick H. Worley is a distinguished research computer scientist in the Computer Science and Mathematics Division of Oak Ridge National Laboratory. His research interests include parallel algorithm design and implementation (especially as applied to simulation models used in climate, fusion energy, and astrophysics research) and the performance evaluation of parallel applications and computer systems. He is currently the co-chair for the performance group for the Accelerated Climate Modeling for Energy project. Pat has a Ph.D. in computer science from Stanford University. He is a member of the Association for Computing Machinery and of the Society for Industrial and Applied Mathematics.

Richard K. Archibald is a staff scientist in the Computational Mathematics group at Oak Ridge National Laboratory. Archibalds current research interests include developing uncertainty quantification algorithms for climate models, designing algorithms for the next generation of high-performance architecture, and establishing long-time integration steps for the dynamical core of climate models at high resolution. Archibald received both his BS in physics and his MS in mathematics from the University of Alberta in Edmonton, Canada in 1996 and 1998, respectively. He obtained his PhD in mathematics from the University of Alberta-Edmonton in 2002.

Matthew R. Norman is a staff scientist in the Scientific Computing Group at the Oak Ridge National Laboratory since 2011. Norman currently researches novel low-communication and large time-step spatial operators and time-explicit temporal operators, porting of climate model codes to utilize GPUs using the OpenACC specification, and performance optimization on HPC in general. Norman has worked with various domain codes including climate, seismology, and CFD. Norman received his B.S. in Meteorology and B.S. in Computer Science (2007), M.S. in Atmospheric Science (2009), and Ph.D in Atmospheric Science (2011) at North Carolina State University. Norman is currently on-node performance task lead for the Accelerated Climate Model for Energy (ACME) project.

Carol S. Woodward is a senior computational scientist in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. Her research interests include iterative solvers for nonlinear and linear systems, time integration methods, and portable numerical software development. Dr. Woodward received a BS in Mathematics from Louisiana State University and a PhD in Computational and Applied Mathematics from Rice University. She has been at LLNL since 1996. Dr. Woodward serves as Chair of the Society for Industrial and Applied Mathematics Activity Group on Geosciences and is also a member of the IEEE, the American Geophysical Union, and the Association for Women in Mathematics.

David J. Gardner is a computational scientist in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. His research interests center on developing accurate and efficient numerical methods, including nonlinear solvers, preconditioners, and fully

implicit/implicit-explicit integrators, for multiscale and multiphysics models. David received his Ph.D. in computational and applied mathematics from Southern Methodist University in 2014. He is a member of the Society for Industrial and Applied Mathematics.

Mark Taylor is a mathematician who specializes in numerical methods for parallel computing and geophysical flows. He currently serves as Chief Computational Scientist for the DOE's Accelerated Climate Modeling for Energy (ACME) project. Taylor developed the mimetic/conservative formulation of the spectral element method, one of the atmospheric dynamical cores used in the Community Earth System Model (CESM) and the ACME project. He received his Ph.D. from New York University's Courant Institute of Mathematical Sciences in 1992 and has worked at Sandia National Laboratories since 2004. In 2014 he was awarded (with Drs. David Bader and William Collins) The Secretary of Energy Achievement Award for his work unifying the Department of Energy's climate modeling research community, enabling the development of high-resolution fully-coupled climate-system simulations. Previously he was a co-chair of the CESM Atmosphere Model Working group (2011-2014) and is currently a member of the Korean Institute of Atmospheric Prediction Systems' Science Advisory Committee (2013-2015).

References

R. K. Archibald, K. J. Evans, and A.G. Salinger. Accelerating time integration for climate modeling using GPUs. Procedia Computer Science, 51:2046–2055, 2015.

E. Bavier, M. Hoemmen, S. Rajamanickam, and H. Thornquist. An overview of the Trilinos project. Scientific Programming, 20(3), 2012.

J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. Math. Comp., 30:772–795, 1976.

J. Dennis, J. Edwards, K. J. Evans, O. Guba, P.H. Lauritzen, A. Mirin, A. St.-Cyr, M.A. Taylor, and P. H. Worley. A scalable spectral element dynamical core for the Community Atmosphere Model. Internat. J. High Perf. Comput. Appl., 26:74–89, 2012. doi: 10.1007/978-3-642-01973-9.

S. Dharmaraja. An analysis of the TR-BDF2 integration scheme. Master's thesis, Massachusetts Institute of Technology. Computation for Design and Optimization Program, 2007.

H. C. Edwards, C.R. Trott, and D. Sunderland. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. J. Parallel and Distributed Computing, 74:363–370, 2012.

D. Estep, V. Ginting, D. Ropp, J. N. Shadid, and S. Tavener. An a posteriori-a priori analysis of multiscale operator splitting. SIAM J. Numer. Anal., 46:1116–1146, March 2008. doi: 10.1137/07068237X.

K. J. Evans, M. A. Taylor, and J. B. Drake. Accuracy analysis of a spectral element atmospheric model using a fully implicit solution framework. Mon. Wea. Rev., 138:3333–3341, 2010.

K.J. Evans, D.W.I. Rouson, A.G. Salinger, M.A. Taylor, J. B. White III, and W. Weijer. A scalable and adaptable solution framework for components of the Community Climate System Model. Lecture Notes in Comp. Sci., 5545:332–341, 2009. doi: 10.1007/978-3-642-01973-9.

K.J. Evans, P. Lauritzen, R. Neale, S. Mishra, M. A. Taylor, and J. Tribbia. AMIP simulation with the CAM4 spectral element dynamical core. J. Climate, 26:689–709, 2013. doi: 10.1175/JCLI-D-11-00448.1.

V. Frayssé, L. Giraud, and H. Kharraz-Aroussi. On the influence of the orthogonalization scheme on the parallel performance of gmres. In David Pritchard and Jeff Reeve, editors, Euro-Par'98 Parallel Processing: 4th International Euro-Par Conference Southampton, UK, September 1–4, 1998 Proceedings, pages 751–762. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-49920-6. doi: 10.1007/BFb0057927.

J. Galewsky, R. K. Scott, and L. M. Polvani. An initial-value problem for testing numerical models of the global shallow-water equations. Tellus, 56A:429–440, 2004.

O. Guba, M. A. Taylor, P. A. Ullrich, J.R. Overfelt, and M. N. Levy. The spectral element method (SEM) on variable-resolution grids: evaluating grid sensitivity and resolution-aware numerical viscosity. Geophys. Model Dev., 7:2803–2816, 2014. doi: 10.5194/gmd-7-2803-2014.

M. A. Heroux, R. A. Bartlett, V. E. Howe, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, and A. Williams. An overview of the Trilinos project. ACM Trans. Math. Soft., 31(3):397–423, 2005.

C. Jablonowski and D. L. Williamson. Diffusion, filters, and fixers in atmospheric general circulation models. In P. H Lauritzen, C. Jablownowski, M. A. Taylor, and R. D. Nair, editors, Numerical Techniques for Global Atmospheric Models, pages 381–493. Spinger-Verlag, Berlin, 2013.

R. Jakob, J. J. Hack, and D. L. Williamson. Spectral transform solution of the shallow water test set. J. Comput. Phys., 119:164–187, 1995.

J. Jia, J.C. Hill, K.J. Evans, G. I. Fann, and M. A. Taylor. A spectral deferred correction method applied to the shallow water equations on a sphere. Mon. Wea. Rev., pages 3435–3449., 2013.

T. Jung, M. J. Miller, T.N. Palmer, P. Towers, N. Wedi, D. Achuthavarier, J.M. Adams, E. L. Altshuler, B. A. Cash, J. L. Kinter, III, L. Marx, C. Stan, and K. I. Hodges. High-resolution global climate simulations with the ECMWF Model in Project Athena: Experimental Design, Model Climate, and Seasonal Forecast Skill. J. Climate, 25:3155–3172, 2012. doi: 10.1175/JCLI-D-11-00265.1.

D. A. Knoll, L. Chacon, L. Margolin, and V. Mousseau. On balanced approximations for time integration of multiple time scale systems. J. Comput. Phys., 185:583–611, 2003.

D.A. Knoll and D.E. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. J. Comput. Phys., 193:357–397, 2004.

P.A. Lott, C.S. Woodward, and K.J. Evans. Algorithmically scalable block preconditioner for fully implicit shallow water equations in CAM-SE. Comp. Geosci., 19:49–61, 2015. doi: 10.1007/s10596-014-9447-6.

S. Mishra and S. Sahany. Effects of time step size on the simulation of tropical climate in NCAR-CAM3. Climate Dynamics, 37:689–704, 2011. doi: doi:10.1007/s00382-011-0994-4.

R. B. Neale, C.-C. Chen, A. Gettleman, P.H. Lauritzen, S. Park, D.L. Williamson, A.J.Conley, R. Garcia, D. Kinnison, J.-F. Francois, D. Marsh, K. Mills, A.K. Smith, S. Tilmes, H. Morrison, P. Cameron-Smith, W. D. Collins, M. J. Iacono, R. Easter, S.J. Ghan, X. Lia,

P.J. Rasch, and M.A. Taylor. Description of the Community Atmosphere Model (CAM 5.0). NCAR Technical Note, TN-486+STR, 2010.

M.R. Norman, , A. Vose J. Larkin, and K.J. Evans. A case study of CUDA FORTRAN and OpenACC for an atmospheric climate kernel, implicit climate dynamics. J. Comp. Sci., 9: 1–6, 2015.

M. Rancic, R. Purser, and F. Mesinger. A global shallow-water model using a expanded spherical cube:Gnomic versus conformal coordinates. Quart. J. Roy. Meteorol. Soc., 122:959–982, 1996.

J. A. Åström, A. Carter, J. Hetherington, K. Ioakimidis, E. Lindahl, G. Mozdzynski, R. W. Nash, P. Schlatter, A. Signell, and J. Westerholm. Preparing scientific application software for exascale computing. In Proceedings of the 11th international conference on Applied Parallel and Scientific Computing. IEEE, 2012. doi: DOI:10.1007/978-3-642-36803-5_2.

A. Robert. The integration of a low order spectral form of the primitive meteorological equations. J. Meteorol. Soc. Japan, 44:237–245, 1966.

M. J. Roberts, P.L. Vidale, M. S. Mizielinski, M. Demory, R. Schiemann, J. Strachan, K. Hodges, R. Bell, and J. Cam. Tropical cyclones in the UPSCALE ensemble of high-resolution global climate model. J. Climate, 28:574–596, 2015. doi: DOI:10.1175/JCLI-D-14-00131.1.

Y. Saad and M. H. Schultz. GMRES: Generalized minimum residual algorithm for solving nonsymmetric systems. SIAM J. Sci. Stat. Comput., 7:856–869, 1986.

A. Sabne, P. Sakdhnagool, S. Lee, and J. Vetter. Evaluating performance portability of OpenACC. In LCPC14, 2014.

S. Sandbach, J. Thurburn, D. Vassilev, and M. G. Duda. A semi-implicit version of the MPAS-Atmosphere dynamical core. Mon. Wea. Rev., 143:3838–3854, 2015.

R. J. Small, J. Bacmeister, D. Bailey, A. Baker, S. Bishop, F. Bryan, J. Caron, J. Dennis, P. Gent, H. Hsu, M. Jochum, D. Lawrence, E. Muoz, P. diNezio, T. Scheitlin, R. Tomas, J. Tribbia, Y. Tseng, and M. Vertenstein. A new synoptic scale resolving global climate simulation using the community earth system model. Journal of Advances in Modeling Earth Systems, 6(4):1065–1094, 2014. ISSN 1942-2466. doi: 10.1002/2014MS000363.

M. A. Taylor and A. Fournier. A compatible and conservative finite element method on unstructured grids. J. Comput. Phys., 229:5879–5895, 2010. doi: 10.1007/978-3-642-01973-9.

M. A. Taylor, J. Tribbia, and M. Iskandarani. The spectral element method for the shallow water equations on the sphere. J. Comput. Phys., 130:92–108, 1997.

M. A. Taylor, J. Edwards, S. Thomas, and R. Nair. A mass and energy conserving spectral element atmospheric dynamical core on the cubed-sphere grid. J. Phys. Conf. Series, 78: 012024, 2007.

M. A. Taylor, J. Edwards, and A. St.-Cyr. Petascale atmospheric models for the Community Climate System Model: New developments and evaluation of scalable dynamic cores. J. Phys. Conf. Series, 125:012023, 2008.

R. Vandewalle, R. Van Driessche, and R. Piessens. The parallel performance of standard parabolic marching schemes. Int. J. High Speed Comp., 3:1–29, 1991.

H. Wan, P. J. Rasch, M.A. Taylor, and C. Jablonowski. Short-term time step convergence in a climate model. J. Adv. Mod. Earth Sys., 7:215–225, 2015.

D. Williamson. The effect of time steps and time-scales on parametrization suites. Quart. J. Roy. Meteorol. Soc., 139:548–560, 2013. doi: 10.1002/qj.1992.

D. L. Williamson, J. B. Drake, J. J. Hack, R. J. Jakob, and P.N. Swarztrauber. A standard test set for numerical approximations to the shallow water equations in spherical geometry. J. Comput. Phys., 102:211–224, 1992.

C. S. Woodward, D. J. Gardner, and K. J. Evans. On the use of finite difference matrix-vector products in newton-krylov solvers for implicit climate dynamics with spectral elements. Procedia Computer Science, 51:2036–2045, 2015.

C. Yang, J. Cao, and X.-C. Cai. A fully implicit domain decomposition algorithm for shallow water equations on the cubed sphere. SIAM J. Sci. Comput., 32:418–438, 2010.

C. Zarzycki, C. Jablonowski, D. R. Thatcher, and M. Taylor. Assessing the model climatology of a multidecadal variable-resolution global resolution atmospheric general circulation model simulation. J. Adv. Mod. Earth Sys., 6:805–828, 2014. doi: doi:10.1002/2014MS000352.