Improving the Scalability of the Ocean Barotropic Solver in the Community Earth System Model

Yong Hu[†], Xiaomeng Huang[†], Allison H. Baker[¶],
Yu-heng Tseng[*], Frank O. Bryan[*], John M. Dennis[¶], Guangwen Yang[†]

[†] Center for Earth System Science
Tsinghua University, 100084, and
Joint Center for Global Change Studies
Beijing, 100875, China
{huyong11,hxm,ygw}@tsinghua.edu.cn

[*]Climate and Global Dynamics Division
[¶]Computational and Information Systems Laboratory
National Center for Atmospheric Research
Boulder,CO. USA
{abaker,ytseng,bryan,dennis}@ucar.edu

ABSTRACT

High-resolution climate simulations are increasingly in demand and require tremendous computing resources. In the Community Earth System Model (CESM), the Parallel Ocean Model (POP) is computationally expensive for high-resolution grids (e.g., 0.1°) and is frequently the least scalable component of CESM for certain production simulations. In particular, the modified Preconditioned Conjugate Gradient (PCG), used to solve the elliptic system of equations in the barotropic mode, scales poorly at the high core counts, which is problematic for high-resolution simulations. In this work, we demonstrate that the communication costs in the barotropic solver occupy an increasing portion of the total POP execution time as core counts are increased. To mitigate this problem, we implement a preconditioned Chebyshev-type iterative method in POP (called P-CSI), which requires far fewer global reductions than PCG. We also develop an effective block preconditioner based on the Error Vector Propagation Method to attain a competitive convergence rate for P-CSI. We demonstrate that the improved scalability of P-CSI results in a 5.2x speedup of the barotropic mode in high-resolution POP on 16,875 cores, which yields a 1.7x speedup of the overall POP simulation. Further, we ensure that the new solver produces an ocean climate consistent with the original one via an ensemble-based statistical method.

Categories and Subject Descriptors

J.2 [**Physical Sciences and Engineering**]: Earth and atmospheric sciences; D.1.3 [**Programming Techniques**]: Parallel Programming

Keywords

parallel computing, linear solver, ocean modeling

SC '15, November 15-20, 2015, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2807591.2807596

1. INTRODUCTION

High-resolution global climate models have become increasingly important in recent years as a means for understanding climate variability and projecting future climate change. The Community Earth System Model (CESM), whose development is centered at the National Center for Atmospheric Research (NCAR), is one of the most widely used global climate models, and its climate projections are a key component in the Intergovernmental Panel on Climate Change (IPCC) Fifth Assessment Report (AR5) [35].

CESM is a fully-coupled climate system model, including atmosphere, ocean, sea-ice and land components. In particular, the ocean component is required to represent processes across a broad range of spatial and temporal scales relevant to climate science. Ocean mesoscale eddies have spatial scales of $\mathcal{O}$(10 - 100 km), one to two orders of magnitude smaller than the dynamically analogous weather systems in the atmosphere. The adjustment time scale for the deep ocean is many centuries up to a few millennia, again orders of magnitude longer than the corresponding timescales in the atmosphere. The computational burden of a global eddy-resolving ocean climate model [6, 25, 17] is thus increased over that for an atmosphere model by the demand for finer spatial resolution and longer integration times.

Moreover, climate model simulations are often run for decades or even centuries, but these long-term simulations are typically too computationally expensive to run at high-resolution. For example, most CESM simulations in IPCC AR5 are carried out with a nominal 1° ocean and a 1° to 2° atmosphere model. Recent increases in both supercomputing resources and high-resolution satellite observations have motivated efforts to improve the parallel performance of high-resolution climate models so that they can be run more routinely (and for less cost).

The Parallel Ocean Model (POP) component of CESM solves the three-dimensional primitive equations with hydrostatic and Boussinesq approximations [34] and divides the time integration into two parts: the baroclinic and the barotropic modes. The baroclinic mode describes the three-dimensional dynamic and thermodynamics processes, and the barotropic mode solves the vertically-integrated momentum and continuity equations in two dimensions. The implicit free-surface method is a common choice in barotropic mode in ocean models because it allows a large time step to efficiently compute the fast gravity mode. However, this

method requires solving a large elliptic system of equations, which scales poorly in POP. In fact, the poor scaling performance of the barotropic solver in POP, which is dominated by the communication overhead [41], is well known, and its optimization will benefit the entire CESM model [11].

The currently recommended linear solver for the barotropic mode in CESM POP is the Chronopoulos-Gear (ChronGear) method [9], a modified Preconditioned Conjugate Gradient method (PCG), combined with a diagonal preconditioner. The required global reduction in the ChronGear method does not scale well and causes a bottleneck for high-resolution simulations. To improve the scaling of POP, and, therefore CESM, we focus on optimizing the barotropic solver by eliminating global reductions and developing a more effective preconditioner. In particular, we make the following contributions:

- We develop a new block parallel preconditioner based on the Error Vector Propagation (EVP) method [31] designed to improve solver convergence in the POP barotropic mode.
- We add a preconditioning interface to the Classical Stiefel Iteration (CSI) solver explored in [20] and implement the resulting preconditioned CSI (P-CSI) solver and new EVP block preconditioner in CESM1.2.0 POP.
- We demonstrate an improvement in convergence rate for both ChronGear and P-CSI when using block EVP.
- We obtain a 5.2x speedup of the barotropic mode in high-resolution POP due to the improved scalability of P-CSI with block EVP preconditioning, greatly improving the scalability of POP (and ultimately CESM) at large core counts.
- We develop and apply an ensemble-based statistical method to evaluate the impact of changing the linear solver in POP and ensure that a consistent ocean climate is produced.

The remainder of this paper is organized as follows. Section 2 discusses POP's barotropic solver and its scalability. Sections 3 and 4 detail the design of P-CSI for POP and the development of the block EVP preconditioner. Section 5 compares the scalability of the ChronGear and P-CSI solvers. Section 6 verifies the new P-CSI solver using the ensemble based statistical method. Finally, related work and conclusions are presented in Sections 7 and 8, respectively.

2. BAROTROPIC SOLVER

The most time-consuming portion of the barotropic simulation is the solution of the elliptic system for the sea surface height (SSH) due to the implicit free-surface algorithm [21]. The implicit elliptic equations for SSH in POP can be expressed as follows:

$$[\nabla \cdot H\nabla - \phi(\tau)]\eta^{n+1} = \psi(\eta^n, \eta^{n-1}, \tau) \quad (1)$$

where H is the depth of the ocean, τ is the time step and η^n is the SSH at the n-th time step, and ψ represents a function of the influence which previous states of SSH and forcing have on the next state. Equation (1) is discretized on a two-dimensional orthogonal curvilinear grid using a nine-point stencil in POP. The stencil can be reorganized into a linear system $Ax = b$.

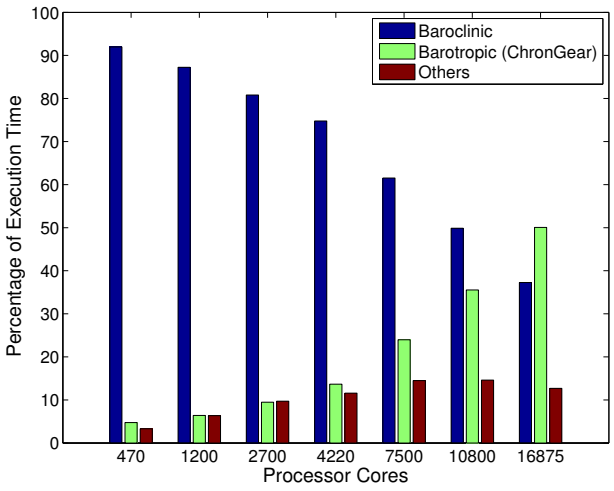

Figure 1: Percentage of execution time in 0.1° POP using the default diagonal-preconditioned ChronGear solver.

POP divides the global domain into blocks and distributes them to processes. Each process only computes the evolution procedures related to the grid points in its own block, and maintains a halo region to update data with its neighbors. We assume that the global domain is size $\mathcal{N} \times \mathcal{N}$ and is divided into $m \times m$ blocks with size of $n \times n$ ($n = \mathcal{N}/m$). We define B and $\tilde{x}$ to be the coefficient matrix and vector associated with a given block (i.e., a sub-matrix of A of size $n^2 \times n^2$), respectively, and the the matrix-vector product of $B\tilde{x}$ requires $9n^2$ operations [20].

2.1 ChronGear solver

ChronGear [9] is a modified conjugate gradient method that combines two global reductions per iteration into one. However, as previously mentioned, when thousands of cores are used in high-resolution (0.1°) POP, the global reduction needed by the inner product in ChronGear is still a major bottleneck. This impact on performance is demonstrated in Figure 1 for 0.1° POP. The percentage execution time for the linear solver increases as the number of processor cores increases. When 470 cores are used, the execution time of the barotropic solver is about 5% of the core POP execution time (excludes initialization and I/O), whereas the baroclinic mode time is close to 90%. However, when several thousand cores are used, the percentage of time spent in the baroclinic decreases, while the percentage of time in the barotropic solver increases. With over sixteen thousand cores, the percentage of the total execution time due to the barotropic solver is nearly 50%.

For reference, the ChronGear method is provided in Algorithm 1. ChronGear contains three major parts: computation, boundary updating, and global reduction. Computation involves matrix-vector and vector-vector multiplications and vector scaling, all of which exhibit good scalability. The cost of the boundary communication, which is required to update the halo area after the matrix-vector multiplication, is bounded and not problematic at the target core counts. We will however illustrate in the subsequent section, that cost of the global reduction, which is required by the inner product in step 9 does however become problematic at large core counts.

Algorithm 1 Chronopoulos-Gear Solver

Require: Coefficient matrix $\mathbf{B}$, preconditioner $\mathbf{M}$, initial guess $\mathbf{x}_0$ and $\mathbf{b}$ associated with grid block $B_{i,j}$

// *do in parallel with all processes*

1: $\mathbf{r}_0 = \mathbf{b} - \mathbf{B}\mathbf{x}_0$, $\mathbf{s}_0 = 0$, $\mathbf{p}_0 = 0$; $\rho_0 = 1$,$\sigma_0 = 0$, $k = 0$;
2: **while** $k \leq k_{max}$ **do**
3: $k = k + 1$;
4: $\mathbf{r}'_k = \mathbf{M}^{-1}\mathbf{r}_{k-1}$; /* preconditioning */
5: $\mathbf{z}_k = \mathbf{B}\mathbf{r}'_k$; /* matrix-vector multiplication */
6: $update\_halo(\mathbf{z}_k)$; /* boundary communication */
7: $\tilde{\rho_k} = \mathbf{r}^T_{k-1}\mathbf{r}'_k$;
8: $\tilde{\delta_k} = \mathbf{z}^T_k\mathbf{r}'_k$;
9: $(\rho_k, \delta_k) = global\_sum(\tilde{\rho_k}, \tilde{\delta_k})$; /* global reduction */
10: $\beta_k = \rho_k/\rho_{k-1}$;
11: $\sigma_k = \delta_k - \beta_k^2\sigma_{k-1}$;
12: $\alpha_k = \rho_k/\sigma_k$;
13: $\mathbf{s}_k = \mathbf{r}'_k + \beta_k\mathbf{s}_{k-1}$;
14: $\mathbf{p}_k = \mathbf{z}_k + \beta_k\mathbf{p}_{k-1}$;
15: $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k\mathbf{s}_k$;
16: $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k\mathbf{p}_k$;
17: $convergence\_check(\mathbf{r}_k)$; /* check convergence */
18: **end while**

2.2 Communication bottleneck

Assume $p = m^2$ processes are used, and each process has exactly one grid block (a typical choice for high-resolution POP). Then the total time of the barotropic mode is equal to the execution time of the ChronGear solver on any block. For each solver iteration, we choose $\mathcal{T}_c$, $\mathcal{T}_b$ and $\mathcal{T}_g$ to be the cost of the computation, boundary updating, and global reduction, respectively.

From Algorithm 1, the computational cost, $\mathcal{T}_c$, contains four vector-scaling operations (steps 13, 14 ,15, and 16), two vector-vector multiplication operations for inner products (steps 7 and 8), and one matrix-vector multiplication (step 5). Therefore, $\mathcal{T}_c = (4n^2 + 2n^2 + 9n^2)\theta + \mathcal{T}_p = 15\frac{\mathcal{N}^2}{p}\theta + \mathcal{T}_p$, where θ is the time unit per floating-point operation and $\mathcal{T}_p$ is the cost of preconditioning. For example, $\mathcal{T}_p = \frac{\mathcal{N}^2}{p}\theta$ for a diagonal preconditioner. When the number of processes increases, $\mathcal{T}_c$ decreases and has a lower limit of zero.

Boundary updating occurs in the halo regions for each process, after operations like matrix-vector multiplication and non-diagonal preconditioning, which require one or more boundary layers. Because every process keeps its own block and two extra halo layers in POP, only one boundary update is needed per iteration even when a non-diagonal preconditioner is used. The actual time depends on the network delay and the volume of the halo regions. With a halo size of 2, the volume in each boundary is $2n$ and decreases as the number of processes increases. The total boundary updating time for each iteration is then $\mathcal{T}_b = 4\alpha + (4 \times 2n)\beta = 4\alpha + (\frac{8\mathcal{N}}{\sqrt{p}})\beta$, where α is point-to-point communication latency per message and β is the transfer time per byte (inverse of bandwidth). The boundary updating time also decreases as the number of processes increases but has a lower bound of 4α.

ChronGear contains only one global reduction per iteration which contains a MPI_allreduce and a masking operation to exclude land points, thus the global reduction time satisfies $\mathcal{T}_g = 2\frac{\mathcal{N}^2}{p}\theta + \log p\alpha$ (assuming that a binomial tree approach is used). The cost of the masking operation should decrease with the number of processes p while the cost of the MPI_allreduce should monotonically increase. Note that the global reduction has virtually no data exchange since there are only two numbers from each process. Combining all three components, the execution time of one diagonal preconditioned ChronGear solver step can be expressed as:

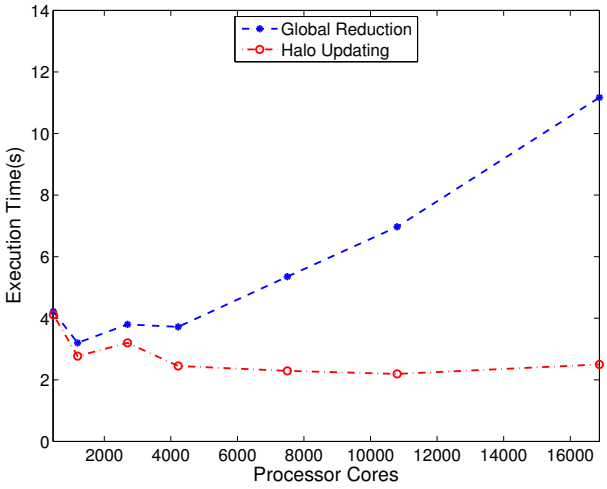

Figure 2: Timing for the global reduction and halo updating components of the ChronGear solver in 0.1° POP for one simulation day on Yellowstone.

$$\begin{aligned}\mathcal{T}_{cg} &= \mathcal{K}_{cg}(\mathcal{T}_c + \mathcal{T}_b + \mathcal{T}_g) \\ &= \mathcal{K}_{cg}[18\tfrac{\mathcal{N}^2}{p}\theta + \tfrac{8\mathcal{N}}{\sqrt{p}}\beta + (4 + \log p)\alpha] \end{aligned} \quad (2)$$

where K_{cg} is the number of iterations, which does not change with the number of processes [20]. Equation (2) shows that the time required for computation and boundary updating decreases as the number of processes increases. But the time required for the global reduction increases with increasing numbers of processes. Therefore, we expect the execution time of the ChronGear solver to increase when the number of processors exceeds a certain threshold. Figure 2 gives timings for the global reduction and boundary (halo) updating components of the ChronGear solver for one simulation day on the Yellowstone machine at NCAR (machine details given in Section 5). Note that the execution time of global reduction becomes dominant and increases when more than a couple thousand cores are used.

3. P-CSI SOLVER

To improve the scalability of POP, a barotropic solver that requires as few global reductions as possible is desired. In [20], Hu et al. proposed an appropriate solver based on Stiefel's CSI method and did a preliminary evaluation at modest core counts in a stand-alone version of POP. Here, we further improve CSI by adding a preconditioning interface, developing an effective preconditioner, and implementing the optimized P-CSI solver into POP within the CESM framework.

The P-CSI algorithm and its properties are similar to those of the CSI algorithm detailed in [20], with the exception of the additional preconditioner (described in detail in the next section). Notably, P-CSI also does not require inner-product operations, potentially improving high core count scalability. The pseudo code for the P-CSI algorithm designed for POP is shown in Algorithm 2. The total computation time for each iteration in the diagonal preconditioned P-CSI solver is $T_c = \frac{12\mathcal{N}^2}{p}\theta + \mathcal{T}_p = \frac{13\mathcal{N}^2}{p}\theta$, and the total

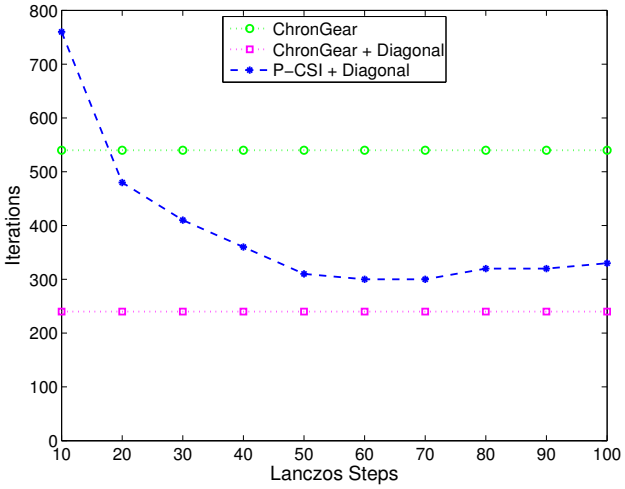

Figure 3: Effect of the number of Lanczos steps on the number of P-CSI iterations and in 1° POP

Algorithm 2 Preconditioned Classical Stiefel Iteration

Require: Coefficient matrix $\mathbf{B}$, preconditioner $\mathbf{M}$, initial guess $\mathbf{x}_0$ and right hand side vector $\mathbf{b}$ associated with grid block $B_{i,j}$; Estimated eigenvalue boundary $[\nu, \mu]$;

// do in parallel with all processes

$\alpha = \frac{2}{\mu-\nu}$, $\beta = \frac{\mu+\nu}{\mu-\nu}$, $\gamma = \frac{\beta}{\alpha}$, $\omega_0 = \frac{2}{\gamma}$; $k = 0$;

$\mathbf{r}_0 = \mathbf{b} - \mathbf{B}\mathbf{x}_0$; $\Delta\mathbf{x}_0 = \gamma^{-1}\mathbf{M}^{-1}\mathbf{r}_0$; $\mathbf{x}_1 = \mathbf{x}_0 + \Delta\mathbf{x}_0$; $\mathbf{r}_1 = \mathbf{b} - \mathbf{B}\mathbf{x}_1$;

while $k \leq k_{max}$ **do**

  $k = k + 1$;

  $\omega_k = 1/(\gamma - \frac{1}{4\alpha^2}\omega_{k-1})$; /* the iterated function */

  $\mathbf{r}'_k = \mathbf{M}^{-1}\mathbf{r}_k$; /* preconditioning */

  $\Delta\mathbf{x}_k = \omega_k\mathbf{r}'_k + (\gamma\omega_k - 1)\Delta\mathbf{x}_{k-1}$;

  $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$;

  $\mathbf{r}_{k+1} = \mathbf{b} - \mathbf{B}\mathbf{x}_{k+1}$; /* matrix–vector multiplication */

  update_halo$(\mathbf{r}_{k+1})$; /* boundary communication */

  convergence_check$(\mathbf{r}_{k+1})$; /* check convergence */

end while

execution time for each P-CSI solve is

$$\mathcal{T}_{pcsi} = \mathcal{K}_{pcsi}(\mathcal{T}_c + \mathcal{T}_b)$$
$$= \mathcal{K}_{pcsi}[13\frac{\mathcal{N}^2}{p}\theta + 4\alpha + \frac{8\mathcal{N}}{\sqrt{p}}\beta] \quad (3)$$

where K_{pcsi} is the number of iterations in one P-CSI solver step.

P-CSI requires approximations of the largest (μ) and smallest (ν) eigenvalues of the preconditioned matrix $M^{-1}A$. Coefficient matrix A and its diagonal preconditioner $M = \Lambda(A)$ are real symmetric positive definite matrices in POP, therefore these extreme eigenvalues can be estimated inexpensively. As in [20] for A, here we use the the Lanczos method [28] for $M^{-1}A$, constructing a series of tridiagonal matrices whose largest and smallest eigenvalues converge to those of $M^{-1}A$. In experiments, we found that setting the Lanczos convergence tolerance ϵ to 0.15 works efficiently in both 1° and 0.1° POP with both diagonal preconditioning and our new block preconditioner. Figure 3 indicates that only a small number of Lanczos steps are necessary to generate eigenvalue estimates of $M^{-1}A$ that result in near-optimal P-CSI convergence. In practice, the cost of the Lanczos method is similar to calling the ChronGear solver a few times.

In contrast with the ChronGear iteration, the P-CSI iteration requires no global reduction except for checking convergence. We note that P-CSI requires a larger number of iterations than ChronGear ($K_{pcsi} > K_{cg}$) in order to obtain the same convergence criteria. We expect that this will translate into a higher execution time for P-CSI than ChronGear at smaller core counts when global reductions are not an issue. However, for high-resolution grids when many cores are required, P-CSI should be notably faster than ChronGear per iteration (see Equations (2) and (3)), which would result in a reduction in time to convergence.

4. A BLOCK-EVP PRECONDITIONER

The total execution time of the barotropic solver is the product of the number of iterations and the execution time per iteration. With increasing numbers of cores, the execution time of computation in each iteration decreases, but the execution time of communication increases. To reduce communication costs, preconditioning is commonly used to reduce the number of iterations to convergence, assuming the cost of preconditioning is reasonable. While the current ChronGear solver in POP has benefited greatly by using a simple diagonal preconditioner [29, 30], further improvement to its convergence rate would significantly reduce the associated communication cost and improve scalability. In fact, the performance of both P-CSI and the existing ChronGear solvers could be further improved with a more effective preconditioner.

4.1 Block preconditioning

Some parallelizable preconditioning methods such as polynomial, approximate-inverse, multigrid, and block preconditioning have drawn much attention recently. High-order polynomial preconditioning can reduce iterations as effectively as incomplete LU factorization (ILU) and its variants [4] in sequential simulations. However, the computational overhead for polynomial preconditioners typically offsets its superiority to diagonal preconditioning (e.g., [26, 33]), as a kth-order polynomial preconditioner requires k matrix vector multiplications in each iteration. Approximate-inverse preconditioners, while highly parallelizable, require solving a linear system several times larger than the original system [33, 5], which makes it less attractive for POP than a simple diagonal preconditioner. Multigrid is highly scalable and generally effective for linear systems derived from elliptic systems of equations. Recent works indicate that geometric multigrid (GMG) is promising in atmosphere [27] and ocean [24, 22] modeling when uniform grids and simple topography are involved. However, in global ocean models, the presence of complex topography (such as islands, straits, passages and coastal complexities) combined with non-uniform or anisotropic grids result in less than ideal scaling for simple GMG methods [24, 15, 38, 36]. Note that in CESM-POP, general dipole orthogonal grids are used to avoid the polar singularity, and only the ocean part of the earth surface is simulated with masked lands. These choices lead to an elliptic system with variable coefficients defined on an irregular domain with non-uniform grids, making the effective use of GMG non-trivial. The scenario is even worse for the high resolution ocean grid where thousands of islands and narrow passages are not representable at a grid coarsening of even one or two levels (e.g., the Bering Strait). For complex geometries, algebraic multigrid (AMG) is often a viable alternative to GMG. A drawback of AMG, though, is

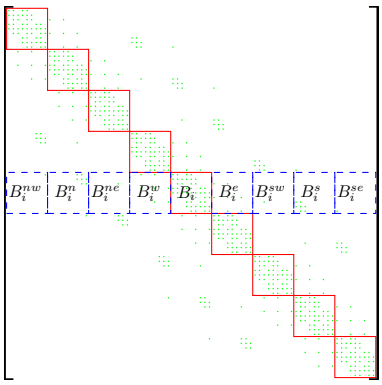

Figure 4: Sparsity pattern of the coefficient matrix developed from nine-point stencils. The whole domain is divided into 3×3 non-overlapping blocks. Elements in red rectangles are coefficients between points in blocks. Elements in blue rectangles are coefficients between points from the i-th block and its neighbor blocks.

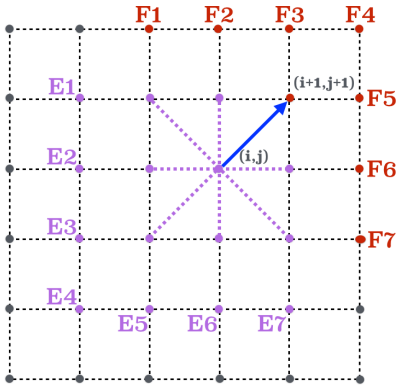

Figure 5: EVP marching method for nine-point stencil. The solution on point $(i+1, j+1)$ can be calculated using the equation on point (i, j), providing solutions on other neighbor points of point (i, j).

that in some cases, the setup cost can exceed that of iterative solver itself, making it inferior to a well-preconditioned CG method (e.g., [27]), particularly when the number of CG iterations is reasonably low as for CESM-POP (Figure 6 in the next section). Further, because POP requires solving the linear equation tens and hundreds of times per simulation day in the low and high resolution versions, respectively, and thousands of simulation years are needed in the typical simulation, the costly setup of AMG is prohibitive. Finally, block preconditioning has been shown to be an effective parallel preconditioner (e.g., [8, 40]) and is appealing for POP because it makes use of the block structure of the coefficient matrix that arises from discretization of the elliptic equations.

To facilitate describing the new block EVP preconditioner, we first briefly review a general block preconditioner, as illustrated by Figure 4. If the linear system of $\mathcal{N} \times \mathcal{N}$ grid points is reordered block-by-block with size of $n \times n$ (e.g., $\mathcal{N}/3 \times \mathcal{N}/3$ in Figure 4), then coefficient matrix A can be represented by a nine-diagonal block matrix. Each row of this matrix contains nine sub-matrices. Each B_i (red blocks) is a block matrix containing coefficients of the grid points in the i-th block, which share the same structure as A but have a smaller size ($n^2 \times n^2$). B_i^e, B_i^w, B_i^n and B_i^s are block matrices containing coefficients of points on east, west, north and south boundaries and the points on their respective neighboring blocks, thus having at most $3n$ nonzero elements distributed on n rows. B_i^{nw}, B_i^{ne}, B_i^{sw} and B_i^{se} have only one nonzero element, representing the coefficient of corner points and their neighboring points on the northwest, northeast, southwest and southeast blocks. The traditional block preconditioning method constructs the approximate inverse of A by sequentially factorizing it with approximations of B_i^{-1}, which is ill-suited for parallel applications. In contrast, the inverse of the block diagonal of A, which provides a good approximation for A, can be calculated naturally in parallel. The inverse of the diagonal block matrices is

$$M^{-1} = \begin{bmatrix} B_1^{-1} & & \\ & \ddots & \\ & & B_{m^2}^{-1} \end{bmatrix}$$

Using M as a preconditioner, the preconditioning process $\mathbf{x} = M^{-1}\mathbf{y}$ is typically transformed into solving the sparse linear equations $B_i\mathbf{x}_i = \mathbf{y}_i$ for each block, (instead of explicitly constructing B_i^{-1}) and solving via LU decomposition. The arithmetic complexity of solving these equations with LU decomposition is $\mathcal{O}(n^4)$, providing that the LU decomposition is previously initialized.

4.2 Error Vector Propagation method

In contrast with LU decomposition, the arithmetic complexity of solving the equations $B_i\mathbf{x}_i = \mathbf{y}_i$ is $\mathcal{O}(n^2)$, where B_i is $n^2 \times n^2$, when using the Error Vector Propagation (EVP) method, which to the best of our knowledge is one of the least costly algorithms for solving elliptic equations in serial [31]. The EVP method and its variants have been used in several ocean models (e.g., Sandia Ocean Modeling System [13] and CANadian version of DIEcast [32]). Further, Tseng and Chien [37] employed a modified parallel EVP method based on domain-decomposition as a solver for the global ocean simulation.

The EVP method works as follows. We discretize Equation 1 (the implicit elliptic system of equations for SSH) into the following form so that we can march the solution northeastward assuming all other neighboring points are exactly known :

$$\begin{aligned}\eta_{i+1,j+1} = (1/A_{i,j}^{ne})(\psi_{i,j} - A_{i,j}^{0}\eta_{i,j} - A_{i,j}^{e}\eta_{i+1,j} \\ -A_{i,j}^{n}\eta_{i,j+1} - A_{i-1,j}^{ne}\eta_{i-1,j+1} + A_{i-1,j}^{e}\eta_{i-1,j} \\ -A_{i-1,j-1}^{ne}\eta_{i-1,j-1} - A_{i,j-1}^{n}\eta_{i,j-1} - A_{i+1,j-1}^{ne}\eta_{i,j-1}) \quad (4)\end{aligned}$$

Figure 5 illustrates a Dirichlet boundary elliptic equation $\mathcal{B}\mathbf{x} = \psi$ on a small domain. We define the interior points next to the south and west boundaries as the initial guess points $\mathbf{e}$ and those next to the north and east boundaries are the final boundary points $\mathbf{f}$ (e.g., $\mathbf{e} = \{E_1, \ldots, E_7\}$, $\mathbf{f} = \{F_1, \ldots, F_7\}$ in Figure 5). If the true solution on $\mathbf{e}$ is known, the exact values over the whole domain can be computed sequentially from southwest to northeast corners, using Equation 4. This procedure is referred to as marching. Unfortunately, the value on $\mathbf{e}$ is often not known until the elliptic equation is solved. However, we can get a solution $\mathbf{x}$ satisfying the elliptic equation on the whole domain except on the boundary, by first guessing the value $\mathbf{x}|_{\mathbf{e}}$ on $\mathbf{e}$ and then calculating the rest using the marching method.

Algorithm 3 Nine-point Error Vector Propagation method

Require: Residual ψ associated with a domain containing $n \times n$ grid points, $k = size(\mathbf{e}) = 2n - 5$;

// *preprocessing*

$\mathbf{x} = \mathbf{0}$

for i = 1, k **do**

$\mathbf{x}|_{\mathbf{e}}(i) = 1$

$\mathbf{x} = marching(\mathbf{x}, \mathbf{0})$

$W(i, :) = \mathbf{x}|_{\mathbf{f}}$

$\mathbf{x}|_{\mathbf{e}}(i) = 0$

end for

$R = inverse(W)$

// *solving*

$\mathbf{x} = marching(\mathbf{x}, \psi)$

$F = (\mathbf{x} - \eta)|_{\mathbf{f}}$

$\mathbf{x}|_{\mathbf{e}} = \mathbf{x}|_{\mathbf{e}} - R * F$

$\mathbf{x} = marching(\mathbf{x}, \psi)$

Then $E = (\mathbf{x} - \eta)|_{\mathbf{e}}$ and $F = (\mathbf{x} - \eta)|_{\mathbf{f}}$ are error vectors on $\mathbf{e}$ and $\mathbf{f}$, respectively. The error vector F is already known since $\mathbf{f}$ are boundary points (Dirichlet boundary condition is imposed). The relationship between the error on initial guess points and the final boundary points can be represented as $F = W * E$. This influence coefficient matrix W can be formed by marching on the whole domain with unit vectors on the initial guess points and zero residual value in the whole domain. We summarized the EVP algorithm for an elliptic equation with zero boundary in Algorithm 3.

The EVP method contains two steps: preprocessing and solving. In the preprocessing step, the influence coefficient matrix and its inverse are computed, involving a calculation of $\mathcal{C}_{pre} = (2n-5)*9n^2+(2n-5)^3 = \mathcal{O}(26n^3)$. Obtaining the solution in the solve step requires $\mathcal{C}_{evp} = 2*9n^2+(2n-5)^2 = \mathcal{O}(22n^2)$. This estimate indicates that EVP has lower computational cost for the solver step than other direct solvers such as LU. Therefore, EVP can be practical in real applications from a cost standpoint because preprocessing is only needed once at the beginning to obtain the influence coefficient matrix and its inverse.

4.3 EVP as a parallel preconditioner

The EVP method described above is an efficient option for solving elliptic equations. However, a major drawback of EVP is that it cannot solve on a large domain without further modifications due to numerical instabilities when marching [31]. But on a small domain up to the size of 12×12, EVP solves with an acceptable round-off error of $\mathcal{O}(10^{-8})$ when double-precision floating-point is used. Its effectiveness on small domains and low-computational cost make EVP an ideal method for parallel block preconditioning. Here, we develop a block preconditioning technique based on the EVP method in each block to further improve the performance of the barotropic solver in POP. Each preconditioner step solves the elliptic equations $B_i\mathbf{x} = \mathbf{y}(i = 1, ..., m^2)$ in parallel.

The fact that EVP is not well-suited for large domains is not an issue for large-scale parallel computing, where larger number of processors result in smaller domains. Furthermore, for our system of equations, the coefficients related to north, south, east and west neighbors on every point are one magnitude order smaller than the others. We found that removing these coefficients reduces the cost of EVP preconditioning by about a half without any significant impact on the convergence rate when used with both ChronGear and P-CSI. As a result, the execution time of EVP preconditioning can be expressed as $\mathcal{T}'_p = 14n^2\theta = 14\frac{\mathcal{N}^2}{p}\theta$. The actual cost of $\mathcal{T}'_p$ depends on the size of the local block, which decreases as more processor cores are used. Thus, the total execution time for one ChronGear and P-CSI solver step with block-EVP preconditioning are

$$\mathcal{T}'_{cg} = \mathcal{K}'_{cg}(\mathcal{T}'_c + \mathcal{T}'_b + \mathcal{T}'_g)$$
$$= \mathcal{K}'_{cg}[31\frac{\mathcal{N}^2}{p}\theta + \frac{8\mathcal{N}}{\sqrt{p}}\beta + (4 + \log p)\alpha], \quad (5)$$

and

$$\mathcal{T}'_{pcsi} = \mathcal{K}'_{pcsi}(\mathcal{T}'_c + \mathcal{T}'_b)$$
$$= \mathcal{K}'_{pcsi}[26\frac{\mathcal{N}^2}{p}\theta + 4\alpha + \frac{8\mathcal{N}}{\sqrt{p}}\beta], \quad (6)$$

respectively.

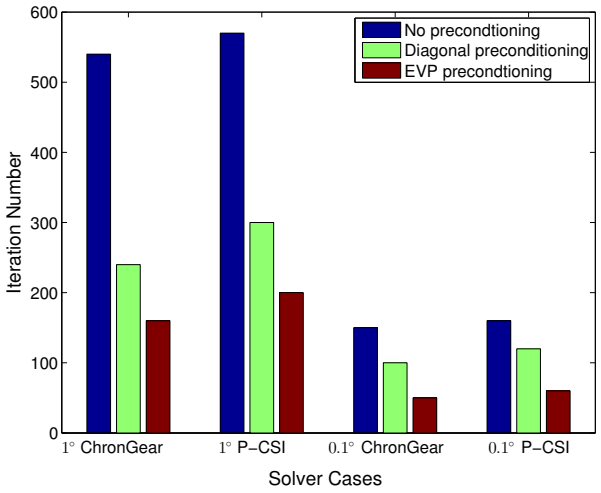

Figure 6: Average number of iterations for different barotropic solvers.

The implementation of EVP preconditioning in POP significantly reduces the number of iterations required for convergence for both the ChronGear and P-CSI solvers. In particular, Figure 6 demonstrates that EVP preconditioning reduces the iteration count by about two-thirds for both the 1° and 0.1 ° resolutions, which is comparable to the approximate-inverse preconditioner proposed in [33] (which is not implemented in CESM POP). Although EVP preconditioning doubles the computation in each iteration, it halves both global and boundary communications which dominate in the barotropic execution time at large core counts. Another advantage of EVP preconditioning is the low preprocessing cost. In 0.1° case, the cost of setting up the preconditioning matrix is less than that of one call to the solver when 512 processor cores are used, and this cost is further decreased when more processors are used. Finally, we note that the 0.1° case requires fewer iterations than the 1° case, because the higher resolution POP grid has a ratio of longitude to latitude grid spacing that is closer to 1, resulting in a smaller condition number for the coefficient matrix.

5. EXPERIMENTAL RESULTS

We first evaluate the performance of our new barotropic solver in CESM1.2.0 on the Yellowstone supercomputer, located at NCAR-Wyoming Supercomputing Center (NWSC) [23]. Yellowstone uses 2.6-Ghz Intel Xeon E5-2670 "Sandy

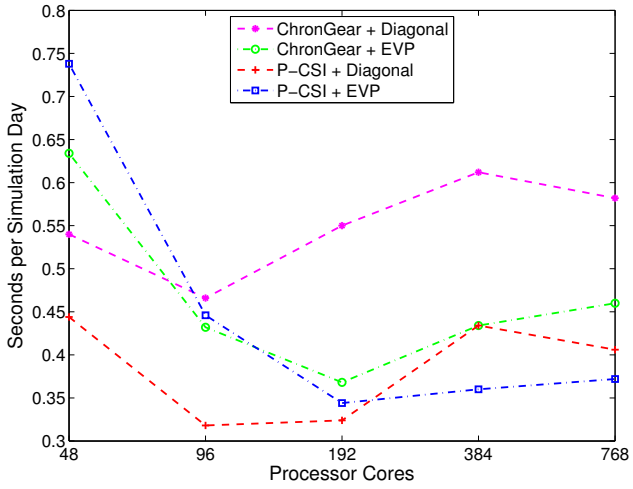

Figure 7: Execution times for the barotropic mode in 1° POP for one simulation day.

Bridge" processors providing a total of 72,576 cores that are connected by a 13.6 GBps InfiniBand network. Obtaining good performance on Yellowstone is critical as its role is to support atmospheric sciences and more than 50% of its usage is due to CESM [42] To focus on the performance of POP, we use the CESM "G_NORMAL_YEAR" component set which uses active ocean and sea ice components (the atmosphere component is data-driven). We examine the two most frequently-used POP horizontal grid resolutions: 1° (320 × 384) and 0.1° (3600 × 2400). Note that by default, CESM1.2.0 sets Yellowstone's MPI environment eager limit (MP_EAGER_LIMIT), which controls the maximum message size before a rendezvous protocol is used, to zero. We discovered that by using the default eager limit on Yellowstone (MP_EAGER_LIMIT = 131072) instead, POP performance significantly improves.

5.1 Low-resolution simulations

The execution times for the barotropic mode with available solvers in 1° POP on Yellowstone are shown in Figure 7. With the default diagonal preconditioning, P-CSI out-performs ChronGear on all core counts and reduces the solver execution time from 0.58s to 0.41s per simulation day (1.4x speedup) at the highest core count (768). Further, with the new block EVP preconditioner, convergence is improved for both the ChronGear and P-CSI solvers at higher core counts. At 768 cores, P-CSI with EVP achieves 0.37s per simulation day, which is an 1.6x improvement over the original ChronGear solver with diagonal preconditioning.

Table 1: Percent improvement of the total execution time for 1° POP on Yellowstone.

Number of cores	48	96	192	384	768
ChronGear+EVP	-.5%	1.1%	6.5%	10.8%	12.1 %
P-CSI+Diagonal	.7%	3.9%	9.3%	11.0%	12.6 %
P-CSI+EVP	-2.4%	.4%	7.4%	14.4%	16.7%

The improvement of the barotropic solver reduces the total execution time for the entire POP model. Table 1 lists the percentage improvement of POP for the three new solver/preconditioner options compared to POP with the diagonal-preconditioned ChronGear solver. Times were obtained from a 5-day simulation, with model initialization and I/O excluded. P-CSI with a block-EVP preconditioner yields a 16.7% improvement on 768 processor cores. While a 16.7% improvement may seem modest, POP at 1° resolution is commonly run for multi-century timescales. Such an improvement may translate into the saving of millions of CPU hours. Further, the 1° resolution needs to be run at (relatively) high core counts when POP is configured with biogeochemistry mode due to the many additional tracers required.

5.2 High-resolution simulations

Now we test the scalability of the new barotropic solver in high-resolution 0.1° POP on Yellowstone. At this resolution, the choice of ocean block size and layout, which affects the distribution of work across processors, has a large impact on performance. Therefore, to remove this influence from our scaling results, we were careful to specify block decompositions for each core count with the same aspect ratio (3:2) and land ratio (.25) and to use space-filling curves. We use the default timestep for 0.1° POP, which is 500 time steps per day (dt_count = 500). Finally, for the sake of consistency, for all solvers we checked for convergence every 10 iterations. Note that because P-CSI iterations are relatively inexpensive (compared to performing the POP convergence check), P-CSI performance may improve if the check for convergence occurs less frequently.

As shown in Figure 8 (left), ChronGear performance begins to degrade after about 2700 cores, while the execution time for P-CSI becomes relatively flat at that point. With diagonal preconditioning, P-CSI accelerates the barotropic mode in 0.1° POP by 4.3x (from 19.0s to 4.4s per simulation day) on 16,875 cores. EVP preconditioning further improves the performance of both ChronGear and P-CSI, resulting in a speedup of the original barotropic mode by 1.4x and 5.2x, respectively. In Section 2, we demonstrated that the original barotropic solver takes an increasing percentage of POP execution time as the number of cores increases. In particular, on 16,875 cores, ChronGear with diagonal preconditioning accounts for about 50% of the total execution time. In contrast, Figure 9 illustrates the improvement of the barotropic mode with the more-scalable EVP preconditioned P-CSI solver, which constitutes only about 16% of the total execution time on 16,875 cores.

Improvement of the barotropic solver benefits the overall performance of POP, especially at large core counts. Simulation rate (simulated years per wall-clock day) is a popular criterion for climate model performance, and here we use the core simulation rate (i.e., the execution time excluding initialization and I/O costs). A simulation rate of 5 simulated years per wall-clock day is considered the minimum rate required to run long term climate simulations [11], and Figure 8 (right) shows that P-CSI can attain higher rates than ChronGear. The EVP-preconditioned P-CSI solver improves the core simulation rate of POP by 1.7x on 16,875 cores, from 6.2 to 10.5 simulated years per wall-clock day.

To illustrate the source of improvement, more detailed timing information for the barotropic solvers is provided in Figure 10. Figure 10 indicates that P-CSI outperforms ChronGear primarily due to fewer global reductions. The reduction in global reductions will also significantly reduce the sensitivity of POP to operating system noise [14] by increasing the time between global synchronization. In addition, the block-EVP preconditioner reduces the boundary

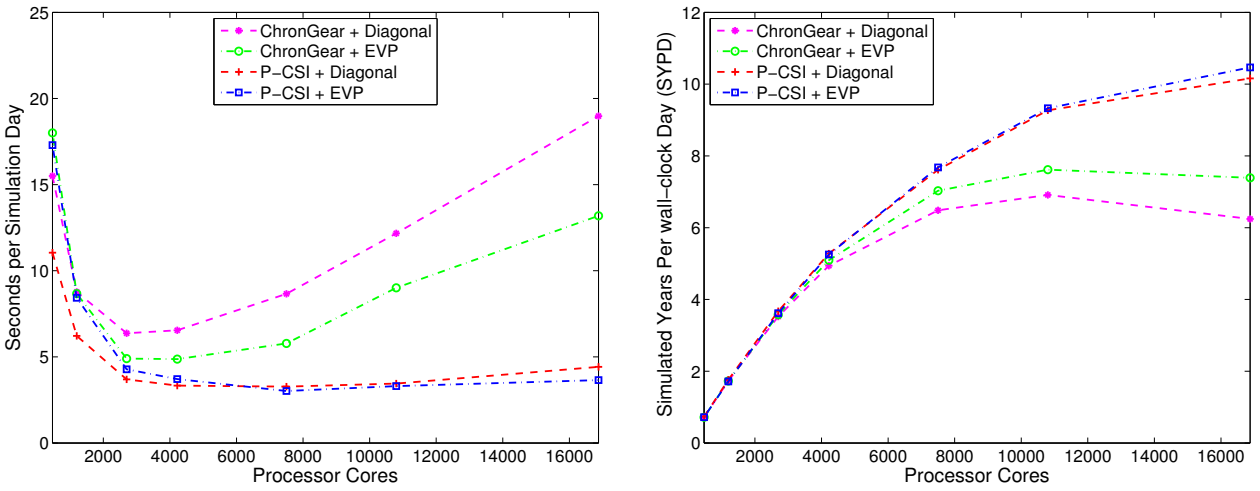

Figure 8: Execution times for the barotropic mode in 0.1° POP for one simulation day on Yellowstone (left). The core simulation rates of 0.1° POP on Yellowstone (right).

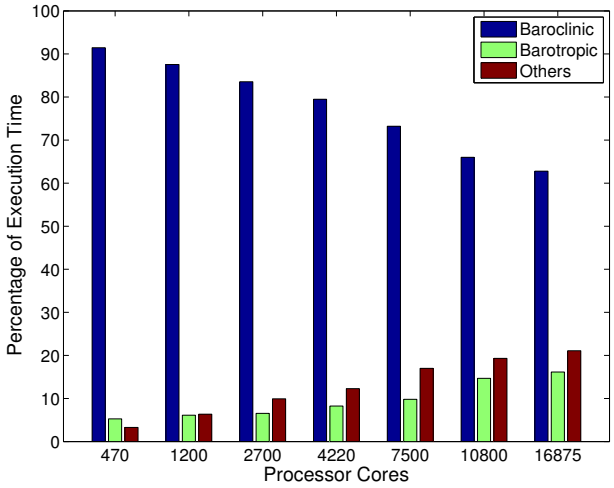

Figure 9: Percentage of execution time in 0.1° POP using P-CSI with block-EVP preconditioning.

update costs by reducing the number of iterations required. Computation costs for the barotropic solver are negligible compared to the global reduction and boundary update costs at large core counts, and, therefore, the extra computations (almost double) required by the EVP preconditioner have little to no impact. Finally, note that the global reduction time actually decreases at less than 1200 cores which is consistent with the theoretical results in equation 2 and 3.

5.3 Simulations on Edison

Now we run the 0.1° POP simulations on the Edison supercomputer to verify that performance improvements are not unique to Yellowstone. Edison, which is the newest supercomputer at the National Energy Research Scientific Computing Center (NERSC), consists of 133,824 2.4 GHz Intel "Ivy Bridge" processor cores connected by an 8GBps Cray Aries high-speed interconnect with Dragonfly topology.

Figure 11 shows that simulations on Edison with the four solver configurations have similar performance characteristics as on Yellowstone. We note that we encountered much more variability in the global communication times in our simulations on Edison (as compared to Yellowstone), likely due to network contention [39]. As a result, the ChronGear times (with both preconditioners) varied a lot from run to run, so we took the average of the best three results to represent the execution time. Because P-CSI has hardly any global reductions (only in the convergence check), the variability in those runs was small. On Edison, P-CSI with diagonal preconditioning in 0.1° POP accelerates the barotropic mode by 3.7x (from 26.2s to 7.0s per simulation day) on 16,875 cores. With EVP preconditioning, both ChronGear and P-CSI performance improves, and the combination of P-CSI and EVP preconditioning results in a 5.6x speedup.

6. EVALUATING THE NEW SOLVER

Due to the chaotic nature of the ocean dynamics, even a round-off difference from the barotropic solver may potentially result in distinct model solutions. Therefore, because we cannot guarantee bit-for-bit (BFB) identical results in ocean solutions when a new solver is introduced, we needed to show that the use of P-CSI with EVP did not result in inaccuracies (or even a changed climate) before it could be formally incorporated into a POP release.

When POP is ported to a new machine, a similar situation occurs where running the same simulation on the two machines is not expected to produce BFB results. The existing POP procedure to verify that a port to a new machine was successful involves running a specific case on the new machine for five simulation days, and then computing the root-mean-square error (RMSE) between the new solution and the standard dataset released by NCAR for the SSH (sea surface height) field. While this procedure provides a simple criterion for evaluating CESM results on new machines (which may contain errors due to the software or hardware environment), we found that it was insufficient for detecting and evaluating solver-induced errors. For example, we ran the 1° case for three years with different convergence tolerances varying from 10^{-10} to 10^{-16} in the barotropic solver (default is 10^{-13}) and calculated the RMSE between a given case and the most strict tolerance case (10^{-16}). Figure 12

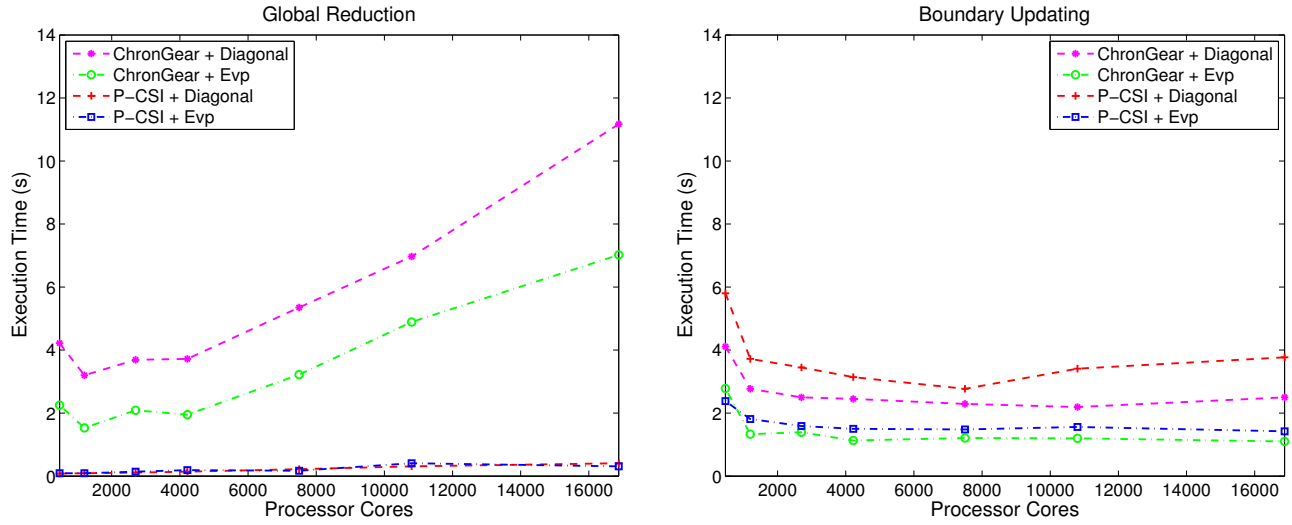

Figure 10: Execution times for the major components of the barotropic solvers in 0.1° POP on Yellowstone: global reduction (left) and boundary communication (right).

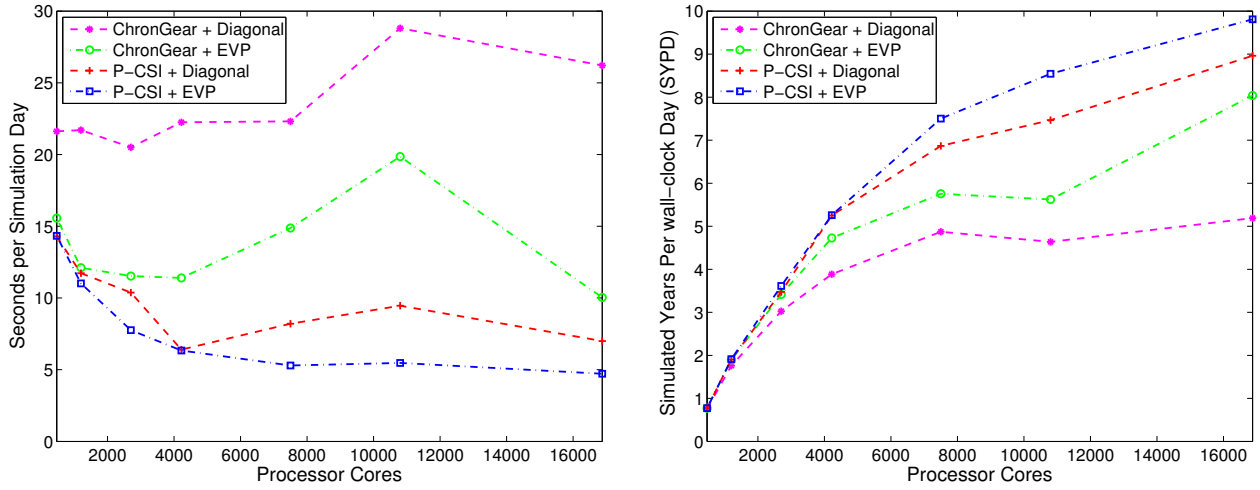

Figure 11: Execution times for the barotropic mode in 0.1° POP for one simulation day on Edison (left). The core simulation rate of 0.1° POP on Edison (right).

shows the RMSE for the temperature field with various convergence tolerances for each month, and clearly error introduced by modifying the solver convergence tolerance is not revealed in the temperature field (nor was it evident in any of the other diagnostic fields, such as velocity and SSH). We had expected that the simulations with tolerances of 10^{-10} and 10^{-11} would have larger RMSE values than the others. However, this was not the case, and, during months twelve and twenty, the 10^{-10} case has almost the smallest RMSE. Note that to isolate the effect of the linear solver, we only looked at error in the open seas (POP does not simulate well on several marginal seas).

Because the existing simple RMSE test was insufficient for detecting whether the climate had been altered, we developed an alternative to evaluating the new ocean solver using a statistical approach. Rather than relying on a single simulation, an ensemble of simulations can better represent the natural variability of the chaotic climate simulation, as described in [2] in the context of data compression for the CESM Community Atmosphere Model (CAM), and be used as a baseline for evaluating non-BFB modifications. Similar to [2], we create an ensemble of simulations which are identical to the default setup except for an order 10^{-14} perturbation in the initial ocean temperature. This perturbation size is not expected to produce different climate model states. We found that an ensemble of size 40 was sufficient for our purposes to represent the variability in the ocean, and we ran longer simulations than for CAM (12-months) due to the longer time-scales present in the ocean. Also note that we ultimately chose to evaluate only the three-dimensional temperature field (instead of the two-dimensional SSH) as we found it to be the most useful diagnostic variable for revealing differences.

We determine whether the new result is consistent with the reference ensemble results as follows. We define the ensemble output at time T as $\mathcal{E} = \{X_1, X_2, ..., X_m\}$, where m is the size of the ensemble. At a given point j, we have a series of possible results for each variable X from the ensemble $\{X_1(j), X_2(j), ..., X_m(j)\}$. As the ensemble size increases, this series more correctly reflects the distribution

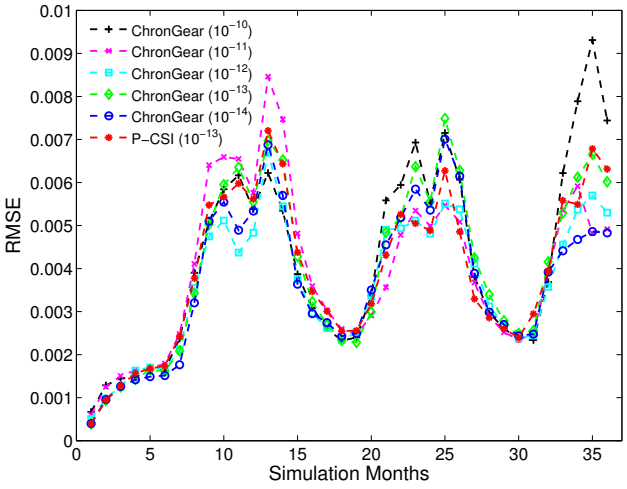

Figure 12: Monthly Root Mean Square Error (RMSE) of temperature for cases with different convergence tolerances in 1° POP.

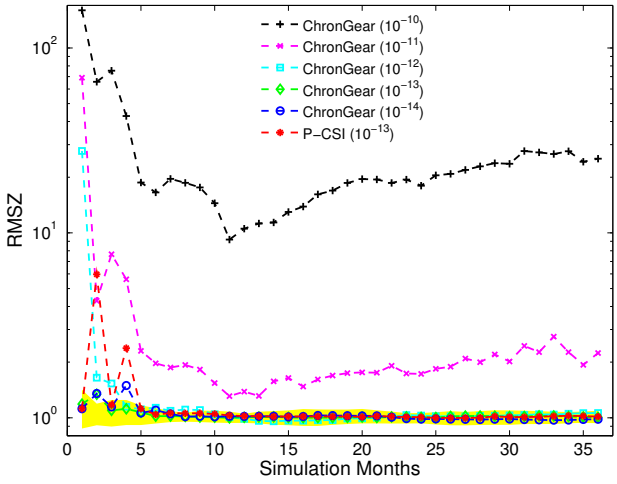

Figure 13: Monthly Root Mean Square Z-score for temperature for cases with different convergence tolerances. The yellow area represent the range of RMSZ within the 40-member ensemble.

of reasonable realization at the given point. We define the mean and standard deviation of this series at point j as $\mu(j)$ and $\delta(j)$, respectively. Let the new case have the result $\tilde{X}$, then the root-mean-square Z-score indicates the average error between the new case and the ensemble data:

$$RMSZ(\tilde{X}, \mathcal{E}) = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(\frac{\tilde{X}(j) - \mu(j)}{\delta(j)})^2}$$

We then re-evaluated the various solver tolerances using the ensemble-based RMSZ measurement. Figure 13 indicates that, unlike the simple RMSE test, the new ensemble-based method is able to identify larger errors due to less strict convergence tolerances. Now the two cases with the loosest tolerances clearly have RMSZ scores on the same order as the error they introduced into the solver and are noticeably removed from the ensemble distribution. This success led us to use the ensemble-based metric to evaluate our new solver and find that the P-CSI results were consist with those of the ensemble (as were the default and stricter tolerances).

7. RELATED WORK

We briefly review related work in two categories: general efforts to improve parallel CG performance and efforts specific to ocean modeling. In the first category, reducing global communication costs for CG has been of interest since the algorithm was parallelized. A particularly nice overview of this effort can be found in [16]. Methods that reduce the number of global reductions over the standard formulation, such as the ChronGear [9] variant used in POP, were early contributions to the field and still popular. Early s-step methods such as that in [7] as well as more recent incarnations (e.g., [19]) also reduce global communications, but combining them with a sophisticated preconditioner is non-trivial. In addition, recent efforts at improving the performance of parallel CG include a variant that overlaps the global-reduction with the matrix-vector computation via a pipelined-approach [16]. We take a different tack along the lines of [18] in that we abandon the CG algorithm and replace it by a simpler iterative method that doesn't include global reductions.

Particular to ocean models, a number of efforts have been made to reduce solver communication overhead. In [41], the use of OpenMP parallelism in the barotropic mode is shown to improve performance at large core counts. Land elimination is another common strategy for reducing communication overhead, and in [10, 12], space-filling curves both improve load-balancing and reduce the number of processes involved in communications by eliminating land blocks. Further, early attempts to overlap communication with computation for parallel ocean general circulation models are proposed in [3], as well as methods to reduce communication by increasing halo sizes. Although all these approaches may improve performance, they do not eliminate the global reduction bottleneck. In fact, the promising preliminary results in our previous work [20], obtained by replacing CG with a Chebyshev type method in the stand-alone variant of POP, encouraged the work in this manuscript.

Finally, our switch away from CG required the development of an effective preconditioner for the barotropic mode. We mention a couple of preconditioning strategies that have been explored to reduce barotropic solver costs on high-resolution grids. Polynomial-preconditioning and local approximate-inverse methods are shown to accelerate CG convergence in a parallel ocean general circulation model in [33]. More recently, an incomplete Cholesky preconditioner was added to the Max Planck Institute ocean model (MPIOM) to improve CG performance on large core counts [1].

8. CONCLUSION

The scalability of high-resolution CESM climate simulations has been impeded by poor performance of the ChronGear barotropic solver in POP at large core counts. This paper improves solver performance by reducing communication costs via an alternative solver with fewer global reductions and by improving convergence via the development of block EVP preconditioner particularly well-suited to the barotropic mode. The performance of the resulting

solver, P-CSI with block-EVP, is evaluated on two machines commonly-used for CESM simulations, and solver speedup is as high as 5x. Confidence that the solver did not adversely impact the ocean simulation was ensured by adapting an ensemble-based consistency strategy to the POP, allowing for the solver's inclusion in a future CESM release. The new barotropic solver will clearly benefit both future low- and high-resolution CESM simulations, particularly for the fully-coupled model whose scalability has historically been inhibited by POP.

9. ACKNOWLEDGMENTS

Computing resources were provided by the Climate Simulation Laboratory at NCAR's Computational and Information Systems Laboratory (sponsored by the NSF and other agencies) and the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

This work is supported in part by a grant from the National Natural Science Foundation of China (41375102), the National Grand Fundamental Research 973 Program of China (No. 2014CB347800), and the National High Technology Development Program of China (2011AA01A203).

10. REFERENCES

[1] P. Adamidis, V. Heuveline, and F. Wilhelm. *A high-efficient scalable solver for the global ocean/sea-ice model MPIOM*. KIT, 2011.

[2] A. H. Baker, H. Xu, J. M. Dennis, M. N. Levy, D. Nychka, S. A. Mickelson, J. Edwards, M. Vertenstein, and A. Wegener. A methodology for evaluating the impact of data compression on climate simulation data. In *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*, pages 203–214. ACM, 2014.

[3] M. Beare and D. Stevens. Optimisation of a parallel ocean general circulation model. In *Annales Geophysicae*, volume 15, pages 1369–1377. Springer, 1997.

[4] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[5] L. Bergamaschi, G. Gambolati, and G. Pini. A numerical experimental study of inverse preconditioning for the parallel iterative solution to 3d finite element flow equations. *Journal of Computational and Applied Mathematics*, 210(1):64–70, 2007.

[6] F. O. Bryan, R. Tomas, J. M. Dennis, D. B. Chelton, N. G. Loeb, and J. L. McClean. Frontal scale air-sea interaction in high-resolution coupled climate models. *Journal of Climate*, 23(23):6277–6291, 2010.

[7] A. T. Chronopoulos and C. W. Gear. S-step iterative methods for symmetric linear systems. *J. Comput. Appl. Math.*, 25(2):153–168, Feb. 1989.

[8] P. Concus, G. Golub, and G. Meurant. Block preconditioning for the conjugate gradient method. *SIAM Journal on Scientific and Statistical Computing*, 6(1):220–252, 1985.

[9] E. D'Azevedo, V. Eijkhout, and C. Romine. Conjugate gradient algorithms with reduced synchronization overhead on distributed memory multiprocessors. 1999.

[10] J. Dennis. Inverse space-filling curve partitioning of a global ocean model. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–10. IEEE, 2007.

[11] J. Dennis, M. Vertenstein, P. Worley, A. Mirin, A. Craig, R. Jacob, and S. Mickelson. Computational performance of ultra-high-resolution capability in the community earth system model. *International Journal of High Performance Computing Applications*, 26(1):5–16, 2012.

[12] J. M. Dennis and H. M. Tufo. Scaling climate simulation applications on the IBM Blue Gene/L system. *IBM Journal of Research and Development*, 52(1.2):117 –126, jan. 2008.

[13] D. E. Dietrich, M. Marietta, and P. J. Roache. An ocean modelling system with turbulent boundary layers and topography: Numerical description. *International journal for numerical methods in fluids*, 7(8):833–855, 1987.

[14] K. B. Ferreira, P. Bridges, and R. Brightwell. Characterizing application sensitivity to OS interference using kernel-level noise injection. In *Proceedings of SC Conference*, pages 1–12. SC Conference, 2008. doi:10.1145/1413370.1413390.

[15] S. R. Fulton, P. E. Ciesielski, and W. H. Schubert. Multigrid methods for elliptic problems: A review. *Monthly Weather Review*, 114(5):943–959, 1986.

[16] P. Ghysels and W. Vanroose. Hiding global synchronization latency in the preconditioned conjugate gradient algorithm. *Parallel Computing*, 40(7):224–238, 2014.

[17] T. Graham. The importance of eddy permitting model resolution for simulation of the heat budget of tropical instability waves. *Ocean Modelling*, 79:21–32, 2014.

[18] M. Gutknecht and S. Röllin. The Chebyshev iteration revisited. *Parallel Computing*, 28(2):263–283, 2002.

[19] M. Hoemmen. *Communication-avoiding Krylov subpace methods*. PhD thesis, University of California, Berkeley, 2010.

[20] Y. Hu, X. Huang, X. Wang, H. Fu, S. Xu, H. Ruan, W. Xue, and G. Yang. A scalable barotropic mode solver for the parallel ocean program. In *Euro-Par 2013 Parallel Processing*, pages 739–750. Springer, 2013.

[21] P. W. Jones, P. Worley, Y. Yoshida, and J. B. White III. Practical performance portability in the parallel ocean program (POP). *Concurrency and Computation: Practice and Experience*, 17:1317–1327, August 2005.

[22] Y. Kanarska, A. Shchepetkin, and J. McWilliams. Algorithm for non-hydrostatic dynamics in the regional oceanic modeling system. *Ocean Modelling*, 18(3):143–174, 2007.

[23] R. Loft, A. Andersen, F. Bryan, J. M. Dennis, T. Engel, P. Gillman, D. Hart, I. Elahi, S. Ghosh, R. Kelly, A. Kamrath, G. Pfister, M. Rempel, J. Small, W. Skamarock, M. Wiltberger, B. Shader, P. Chen, and B. Cash. Yellowstone: A dedicated resource for earth system science. In J. S. Vetter, editor, *Contemporary High Performance Computing: From Petascale Toward Exascale, Volume Two*,

volume 2 of *CRC Computational Science Series*, page 262. Chapman and Hall/CRC, Boca Raton, 1 edition, 2015.

[24] Y. Matsumura and H. Hasumi. A non-hydrostatic ocean model with a scalable multigrid poisson solver. *Ocean Modelling*, 24(1):15–28, 2008.

[25] J. L. McClean, D. C. Bader, F. O. Bryan, M. E. Maltrud, J. M. Dennis, A. A. Mirin, P. W. Jones, Y. Y. Kim, D. P. Ivanova, M. Vertenstein, et al. A prototype two-decade fully-coupled fine-resolution CCSM simulation. *Ocean Modelling*, 39(1):10–30, 2011.

[26] P. D. Meyer, A. J. Valocchi, S. F. Ashby, and P. E. Saylor. A numerical investigation of the conjugate gradient method as applied to three-dimensional groundwater flow problems in randomly heterogeneous porous media. *Water Resources Research*, 25(6):1440–1446, 1989.

[27] E. H. Müller and R. Scheichl. Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction. *Quarterly Journal of the Royal Meteorological Society*, 140(685):2608–2624, 2014.

[28] C. Paige. Accuracy and effectiveness of the lanczos algorithm for the symmetric eigenproblem. *Linear Algebra and its Applications*, 34(0):235 – 258, 1980.

[29] G. Pini and G. Gambolati. Is a simple diagonal scaling the best preconditioner for conjugate gradients on supercomputers? *Advances in Water Resources*, 13(3):147–153, 1990.

[30] R. S. Reddy and M. M. Kumar. Comparison of conjugate gradient methods and strongly implicit procedure for groundwater flow simulation. *Journal of the Indian Institute of Science*, 75(6):667, 2013.

[31] P. J. Roache. *Elliptic marching methods and domain decomposition*, volume 5. CRC press, 1995.

[32] J. Sheng, D. G. Wright, R. J. Greatbatch, and D. E. Dietrich. Candie: A new version of the diecast ocean circulation model. *Journal of Atmospheric and Oceanic Technology*, 15(6):1414–1432, 1998.

[33] R. Smith, J. Dukowicz, and R. Malone. Parallel ocean general circulation modeling. *Physica D: Nonlinear Phenomena*, 60(1):38–61, 1992.

[34] R. Smith, P. Jones, B. Briegleb, F. Bryan, G. Danabasoglu, J. Dennis, J. Dukowicz, C. E. B. Fox-Kemper, P. Gent, M. Hecht, et al. The parallel ocean program (POP) reference manual ocean component of the community climate system model (CCSM). 2010.

[35] T. Stocker, D. Qin, G. Plattner, M. Tignor, S. Allen, J. Boschung, A. Nauels, Y. Xia, B. Bex, and B. Midgley. IPCC, 2013: Climate change 2013: the physical science basis. contribution of working group I to the fifth assessment report of the Intergovernmental Panel on Climate Change. 2013.

[36] K. Stüben. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128(1):281–309, 2001.

[37] Y.-h. Tseng and M.-h. Chien. Parallel domain-decomposed Taiwan multi-scale community ocean model (pd-timcom). *Computers & Fluids*, 45(1):77–83, 2011.

[38] Y.-h. Tseng and J. H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of computational physics*, 192(2):593–623, 2003.

[39] D. Wang, A. Bhatele, and D. Ghosal. Performance variability due to job placement on edison. Poster presented at SC14, Nov 16-21, New Orleans.

[40] J. A. White and R. I. Borja. Block-preconditioned newton–krylov solvers for fully coupled flow and geomechanics. *Computational Geosciences*, 15(4):647–659, 2011.

[41] P. H. Worley, A. A. Mirin, A. P. Craig, M. A. Taylor, J. M. Dennis, and M. Vertenstein. Performance of the community earth system model. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 54:1–54:11, New York, NY, USA, 2011. ACM.

[42] Yellowstone workload study, v4.1. https://www2.cisl.ucar.edu/NWSC-2, September 2014.