A Scalable Barotropic Solver for the Parallel Ocean Program

Yong Hu\textsuperscript{1}, Xiaomeng Huang\textsuperscript{1}, Yu-Heng Tseng\textsuperscript{2}, Allison Baker\textsuperscript{3}, Frank Bryan\textsuperscript{2}, and John Dennis\textsuperscript{3}

\textsuperscript{1} Ministry of Education Key Laboratory for Earth System Modeling
Tsinghua University, China
\textsuperscript{2} Oceanography Section, Climate and Global Dynamics Division
\textsuperscript{3} Computational and Information Systems Laboratory
National Center for Atmospheric Research, USA
POP dominates in CESM

Computational performance of ultra-high-resolution capability in the Community Earth System Model. John Dennis, et al. HPCA’12
Scalability of 0.1º POP

Baroclinic mode time dominates on a small number of cores, Barotropic mode time dominates in a large number of cores (>5000)

Kraken, 99,072 core Cray XT5 system at the National Institute for Computational Science (NICS). Andrew Stone. 2011
Barotropic mode in POP

- Require to solve an elliptic system for SSH in the barotropic mode.
- The elliptic problem is approximated as a linear system.

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

- PCG (ChronGear) is used to solve this linear system.

**Fig. 1.** Grid domain decomposition of POP
Scalability of 0.1° POP on Yellowstone

ChronGear Solver does not scale well on large number of processes.

Yellowstone, at National Center for Atmospheric Research (NCAR). Yong Hu. 2014
ChronGear Solver

Algorithm 1 Chronopoulos-Gear Solver

Require: Coefficient matrix $B$, preconditioner $M$, initial guess $x_0$ and $b$ associated with grid block $B_{i,j}$

// do in parallel with all processes
1: $r_0 = b - Bx_0, s_0 = 0, p_0 = 0; \quad \rho_0 = 1, \sigma_0 = 0, k = 0$
2: while $k \leq k_{\text{max}}$ do
3: \hspace{1em} $k = k + 1$
4: \hspace{1em} $r'_k = M^{-1}r_{k-1}$; /* preconditioning */
5: \hspace{1em} $\text{update}_\text{halo}(r'_{k-1})$; /* boundary communication */
6: \hspace{1em} $z_k = Br'_k$; /* matrix-vector multiplication */
7: \hspace{1em} $\text{update}_\text{halo}(z_k)$; /* boundary communication */
8: \hspace{1em} $\tilde{\rho}_k = r'^T_{k-1}r'_k$
9: \hspace{1em} $\tilde{\delta}_k = z'^T_k r'_k$
10: \hspace{1em} $(\rho_k, \delta_k) = \text{global}\_\text{sum}(\tilde{\rho}_k, \tilde{\delta}_k)$; /* global reduction */
11: \hspace{1em} $\beta_k = \rho_k/\rho_{k-1}$
12: \hspace{1em} $\sigma_k = \delta_k - \beta^2_k \sigma_{k-1}$
13: \hspace{1em} $\alpha_k = \rho_k/\sigma_k$
14: \hspace{1em} $s_k = r'_{k-1} + \beta_k s_{k-1}$
15: \hspace{1em} $p_k = z_k + \beta_k p_{k-1}$
16: \hspace{1em} $x_k = x_{k-1} + \alpha_k s_k$
17: \hspace{1em} $r_k = r_{k-1} - \alpha_k p_k$
18: \hspace{1em} if $k \% n_c == 0$ then
19: \hspace{2em} check convergence;
20: \hspace{1em} end if
21: end while

\[
T_p + 15 \frac{N^2}{p} \theta
\]

$\theta$ : time unit per floating-point operation

$N^2$ : number of grids

$p$ : number of processors
Algorithm 1 Chronopoulos-Gear Solver

Require: Coefficient matrix $B$, preconditioner $M$, initial guess $x_0$ and $b$ associated with grid block $B_{i,j}$

// do in parallel with all processes
1: $r_0 = b - Bx_0, s_0 = 0, p_0 = 0; \quad \rho_0 = 1, \sigma_0 = 0, k = 0$
2: while $k \leq k_{max}$ do
3: \hspace{1em} $k = k + 1$
4: \hspace{1em} $r'_k = M^{-1}r_{k-1};$ /* preconditioning */
5: \hspace{1em} $update\_halo(r'_k);$ /* boundary communication */
6: \hspace{1em} $z_k = Br'_k;$ /* matrix-vector multiplication */
7: \hspace{1em} $update\_halo(z_k);$ /* boundary communication */
8: \hspace{1em} $\tilde{\rho}_k = r'_{k-1}r'_k$
9: \hspace{1em} $\tilde{\delta}_k = z_k^T r'_k$
10: \hspace{1em} $(\rho_k, \delta_k) = global\_sum(\tilde{\rho}_k, \tilde{\delta}_k);$ /* global reduction */
11: \hspace{1em} $\beta_k = \rho_k / \rho_{k-1}$
12: \hspace{1em} $\sigma_k = \delta_k - \beta_k^2 \sigma_{k-1}$
13: \hspace{1em} $\alpha_k = \rho_k / \sigma_k$
14: \hspace{1em} $s_k = r'_{k-1} + \beta_k s_{k-1}$
15: \hspace{1em} $p_k = z_k + \beta_k p_{k-1}$
16: \hspace{1em} $x_k = x_{k-1} + \alpha_k s_k$
17: \hspace{1em} $r_k = r_{k-1} - \alpha_k p_k$
18: \hspace{1em} if $k \% n_c == 0$ then
19: \hspace{2em} check convergence;
20: \hspace{1em} end if
21: end while

$4\alpha + \left( \frac{8N}{\sqrt{p}} \right) \beta$

$\alpha$: communication latency
$\beta$: transfer time per byte
Algorithm 1 Chronopoulos-Gear Solver

Require: Coefficient matrix \( B \), preconditioner \( M \), initial guess \( x_0 \) and \( b \) associated with grid block \( B_{i,j} \)

1: \( r_0 = b - Bx_0, s_0 = 0, p_0 = 0; \quad \rho_0 = 1, \sigma_0 = 0, k = 0; \)
2: while \( k \leq k_{\text{max}} \) do
3: \( k = k + 1; \)
4: \( r'_k = M^{-1} r_{k-1}; \) /* preconditoning */
5: \( \text{update}_\text{halo}(r'_{k-1}); \) /* boundary communication */
6: \( z_k = Br'_k; \) /* matrix-vector multiplication */
7: \( \text{update}_\text{halo}(z_k); \) /* boundary communication */
8: \( \tilde{\rho}_k = r'_{k-1} r'_k; \)
9: \( \tilde{\sigma}_k = z'_k r'_k; \)
10: \( (\rho_k, \sigma_k) = \text{global}_\text{sum}(\tilde{\rho}_k, \tilde{\sigma}_k); \) /* global reduction */
11: \( \beta_k = \rho_k / \rho_{k-1}; \)
12: \( \sigma_k = \sigma_k - \beta_k^2 \sigma_{k-1}; \)
13: \( \alpha_k = \rho_k / \sigma_k; \)
14: \( s_k = r'_{k-1} + \beta_k s_{k-1}; \)
15: \( p_k = z_k + \beta_k p_{k-1}; \)
16: \( x_k = x_{k-1} + \alpha_k s_k; \)
17: \( r_k = r_{k-1} - \alpha_k p_k; \)
18: if \( k \% n_c == 0 \) then
19: \quad check convergence;
20: end if
21: end while

Global Reduction

\[ T_g = \log p \alpha \]
Global reduction is the bottleneck of the ChronGear Solver in 0.1° POP.
Algorithm 2 Preconditioned Stiefel Iteration solver

Require: Coefficient matrix $B$, preconditioner $M$, initial guess $x_0$ and $b$ associated with grid block $B_{i,j}$; Estimated eigenvalue boundary $[\nu, \mu]$;

// do in parallel with all processes

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

2: $r_0 = b - Bx_0$; $x_1 = x_0 - \gamma^{-1}M^{-1}r_0$; $r_1 = b - Bx_1$;

3: while $k \leq k_{max}$ do

4: $k = k + 1$;

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

6: $r'_{k-1} = M^{-1}r_{k-1}$; /* preconditioning */

7: $\Delta x_k = \omega_k r'_{k-1} + (\gamma \omega_k - 1)\Delta x_{k-1}$;

8: $x_k = x_{k-1} + \Delta x_{k-1}$;

9: $r_k = b - Bx_k$; /* matrix–vector multiplication */

10: update_halo($r_k$); /* boundary communication */

11: if $k \% n_c == 0$ then

12: check convergence;

13: end if

14: end while
P-CSI vs. ChronGear

- Unlike ChronGear, P-CSI has no global reduction.
- P-CSI needs slightly more iterations, but less computation in each iteration.
- P-CSI requires two extreme eigenvalues.
Eigenvalue Estimation

- Lanczos method to construct a tridiagonal matrix $T$
- $T$ has eigenvalues close to $M^{-1}A$
- Estimation needed only once, extra overhead less than one barotropic step.
Scalability on Yellowstone

1 degree POP

0.1 degree POP
Error Vector Propagation (EVP) Preconditioning

EVP preconditioning reduces the iteration number to about one-third

<table>
<thead>
<tr>
<th>Resolution</th>
<th>1 degree</th>
<th>0.1 degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solvers</td>
<td>ChronGear</td>
<td>P-CSI</td>
</tr>
<tr>
<td>NONE</td>
<td>540</td>
<td>570</td>
</tr>
<tr>
<td>DIAG</td>
<td>240</td>
<td>300</td>
</tr>
<tr>
<td>EVP</td>
<td>160</td>
<td>200</td>
</tr>
</tbody>
</table>
Scalability on Yellowstone

1 degree POP

0.1 degree POP

Processor Cores

Seconds per Simulation Day

Scalability

0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
48
96
192
384

Processor Cores

Seconds per Simulation Day

Scalability

512
1224
2634
4028
6124
9128

0.1 degree POP

1.4X
3.2X
4.7X
1.8X
1.4X
0.9X

1 degree POP

ChronGear + Diagonal
ChronGear + EVP
P−CSI + Diagonal
P−CSI + EVP

Scalability

1 degree POP

0.1 degree POP
Solver Component

Global Reduction

Halo Updating

Computation

Execution Time (s)

Processor Cores

- ChronGear + Diagonal
- ChronGear + Evp
- P–CSI + Diagonal
- P–CSI + Evp
# Simulated Years for Solver Per wall-clock Day

Simulation rate without considering the I/O and initialization in 0.1 degree POP.

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>512</th>
<th>1224</th>
<th>2634</th>
<th>3476</th>
<th>4028</th>
<th>6124</th>
<th>9128</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChronGear</td>
<td>0.71</td>
<td>1.69</td>
<td>3.31</td>
<td>3.83</td>
<td>4.28</td>
<td>4.72</td>
<td>6.02</td>
</tr>
<tr>
<td>ChronGear+Evp</td>
<td>0.70</td>
<td>1.68</td>
<td>3.42</td>
<td>3.99</td>
<td>4.53</td>
<td>5.08</td>
<td>6.69</td>
</tr>
<tr>
<td>P-CSI+Diagonal</td>
<td>0.72</td>
<td>1.72</td>
<td>3.58</td>
<td>4.36</td>
<td>5.07</td>
<td>6.15</td>
<td>7.75</td>
</tr>
<tr>
<td>P-CSI+Evp</td>
<td>0.71</td>
<td>1.70</td>
<td>3.54</td>
<td>4.36</td>
<td>5.12</td>
<td>6.38</td>
<td>8.10</td>
</tr>
</tbody>
</table>

35%
Q & A

Thanks!