Recent Advances in Sparse Linear Solver Stacks for Exascale

NCAR Multi-core 9 Workshop

Stephen Thomas[1] Kasia Świrydowicz[1] Shreyas Ananthan [1] Michael Sprague [1]
Julien Langou [2] Daniel Bielich [2] Ruipeng Li[3] Rob Falgout[3] Ulrike M. Yang[3]
Mark Hoemmen[4] Ichitaro Yamazaki[4] Eric Boman[4] James Elliot[4]

[1] High Performance Algorithms and Complex Fluids Group
National Renewable Energy Laboratory, Golden CO

[2] Department of Mathematics University of Colorado, Denver, CO

[3] Center for Applied Scientific Computing
Lawrence Livermore National Laboratories, Livermore, CA.

[4] Sandia National Laboratory,
Livermore, CA and Albuquerque, NM

National Center for Atmospheric Research
September 25, 2019

Scalable Solvers

Grand challenge ECP simulations

- High-fidelity wind turbine simulations, to capture wake vortex formation
- Nalu-wind Navier-Stokes incompressible flow solver. $\mathcal{O}(100)$ Billion grid cells
- GMRES with symmetric Gauss-Seidel (SGS), and algebraic multigrid (AMG) preconditioners
- Integration rate of $\mathcal{O}(10)$ sec time per time step. $\mathcal{O}(1)$ hour per revolution

Recent advances in scalable solvers and preconditioners

- Multi-MPI Multi-GPU implementation of low-sync one-reduce ICWY MGS-GMRES (Hypre)
- Multi-MPI Multi-GPU implementation of low-sync one-reduce *s*-step CA-GMRES (Trilinos)
- $2\times$ faster *Nalu-Wind* matrix assembly with optimal *hypre* CSR memory allocations
- low-rank approximate AINV *hypre* smoothers (NVIDIA V100 GPU cuda)
- $25\times$ faster than cuSparse lower triangular solve on GPU

EXASCALE COMPUTING PROJECT

Gram-Schmidt and GMRES

Solve $Ax = b$, iterative Krylov methods in DOE libraries (*hypre*, Trilinos, PETSc)

- Start with x_0 (initial guess), $r_0 = b - Ax_0$, $v_1 = r_0 / \|r_0\|_2$
- Search for update to x_0 in *Krylov space*: $\mathcal{K}_m = span\{\, v_1\,,\, Av_1\,,\, A^2 v_1\,, \ldots A^{m-1} v_1\,\}$,
- Krylov vectors form columns of $V_{:,i}$, with $v_i := V_{:,i}$,
- Arnoldi–GMRES solver based on the *QR* factorization of

$$\begin{bmatrix} r_0, & AV_m \end{bmatrix} = V_{m+1} \begin{bmatrix} \|r_0\| e_1, & H_{m+1,m} \end{bmatrix}$$

Theorem. One synch per column is sufficient for (MGS, CGS-2) Gram-Schmidt *QR* and Arnoldi-*QR*

- Inverse compact *ICWY* MGS-GMRES based on lower-triangular solve, $\mathcal{O}(\varepsilon)\kappa(A)$ orthogonality
- Recursive classical Gram-Schmidt (CGS-2), $\mathcal{O}(\varepsilon)$ orthogonality

Corollary. The backward stability and loss of orthogonality are equivalent to the original algorithms

One-reduce Gram-Schmidt Algorithms

Björck (1967), Lemma 5.1 and Corollary, Ruhe (1983)

Modified Gram-Schmidt projector

$$P^M a_j = (I - Q_{j-1} T^M Q_{j-1}^T) a_j$$

inverse compact WY form $T^M = (I + L_{j-1})^{-1}$

Classical Gram-Schmidt with reorthogonalization

$$P^{IC} a_j = (I - Q_{j-1} T^{IC} Q_{j-1}^T) a_j$$

where we are approximating $P = I - Q_{j-1} (Q_{j-1}^T Q_{j-1})^{-1} Q_{j-1}^T$

$$T^{IC} = (Q_{j-1}^T Q_{j-1})^{-1} = I - L_{j-1} = I - \begin{bmatrix} 0 & 0 \\ -w^T & 0 \end{bmatrix}$$

Inverse Compact WY Modified Gram-Schmidt

$Q^T Q = I + L + L^T$, compute L^T one column per iteration

Algorithm 1 Triangular Solve Modified Gram-Schmidt left-looking (columns)

for $j = 1, 2, \ldots n$ **do**
  $q_j = a_j$
  if $j > 1$ **then**
    $T_{1:j-1,j-1} = Q^T_{:,1:j-1}\, q_{j-1}$ ▷ Synchronization
    $L_{1:j-1,1:j-1} = \mathrm{tril}(T_{1:j-1,1:j-1}, -1)$
    $R_{1:j-1,j} = (I + L_{1:j-1,1:j-1})^{-1}\, Q^T_{:,1:j-1}\, a_j$ ▷ Lower triangular solve
    $q_j = q_j - Q_{:,1:j-1}\, R_{1:j-1,j}$
  end if
  $R_{jj} = \|q_j\|_2$
  $q_j = q_j / R_{jj}$
end for

Algorithm 2 One reduction MGS-GMRES with Lagged Normalization

```
r_0 = b − A x_0,  v_1 = r_0.
v_2 = A v_1
( V_2, R, L_2 ) = mgs( V_2, R, L_1 )
for i = 1, 2, ... do
    v_{i+2} = A v_{i+1}                                          ▷ Matrix-vector product
    [ L^T_{:,i+1}, r_{i+2} ] = V^T_{i+1} [ v_{i+1} v_{i+2} ]     ▷ Global synchronization
    r_{i+1,i+1} = ‖v_{i+1}‖_2
    v_{i+1} = v_{i+1} / r_{i+1,i+1}                              ▷ Lagged normalization
    r_{1:i+1,i+2} = r_{1:i+1,i+2} / r_{i+1,i+1}                  ▷ Scale for Arnoldi
    v_{i+2} = v_{i+2} / r_{i+1,i+1}                              ▷ Scale for Arnoldi
    r_{i+1,i+2} = r_{i+1,i+2} / r_{i+1,i+1}
    L^T_{:,i+1} = L^T_{:,i+1} / r_{i+1,i+1}
    r_{1:i+1,i+2} = ( I + L_{i+1} )^{-1} r_{1:i+1,i+2}           ▷ Lower triangular solve
    v_{i+2} = v_{i+2} − V_{i+1} r_{1:i+1,i+2}
    H_i = R_{i+1}
    Apply Givens rotations to H_i
end for
y_m = argmin‖( H_m y_m − ‖r_0‖_2 e_1 )‖_2
x = x_0 + V_m y_m
```

Normwise Relative Backward Error

Stopping criterion for MGS-GMRES

- When does MGS-GMRES reach minimum error level ?

$$\frac{\|r_k\|_2}{\|b\|_2} = \frac{\|b - Ax_k\|_2}{\|b\|_2} < tol$$

  flattens when $\|S\| = 1$ and the columns of V_k become linearly dependent

- Paige, Rozložnik and Strakoš (2006), normwise relative backwards error

$$\frac{\|r_k\|_2}{\|b\|_2 + \|A\|_2\|x\|_2} < \mathcal{O}(\varepsilon)$$

  achieved when $\|S\|_2 = 1$.

- Paige notes: For a sufficiently nonsingular matrix

$$\sigma_{\min}(A) \gg n^2\varepsilon\|A\|_F$$

  can employ MGS-GMRES to solve $Ax = b$ with NBRE stopping criterion.

Normwise Relative Backward Error

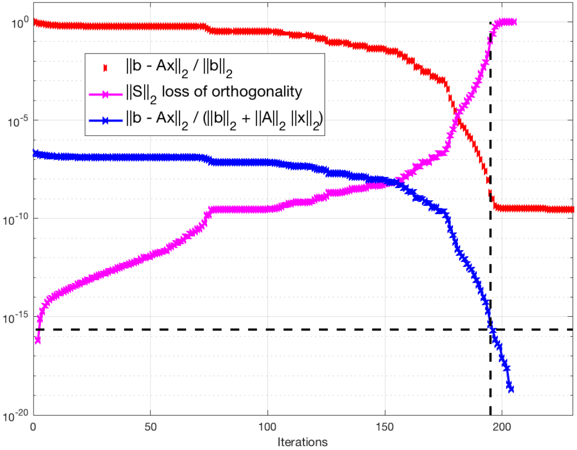
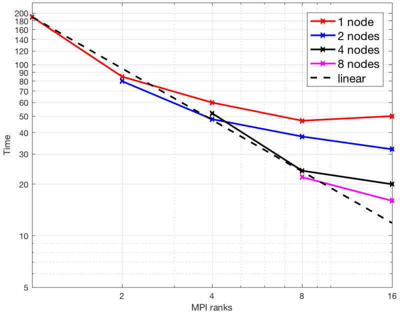


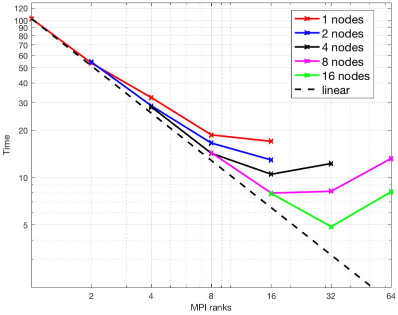
Figure: Greenbaum, Rozložnik and Strakoš (1997). impcol_e matrix

Low-Synch GMRES Multi-MPI Multi-GPU

Świrydowicz et al (2019), Low-Synch Gram-Schmidt and GMRES (NLAA)



(a) Hypre Level-1 BLAS MGS-GMRES



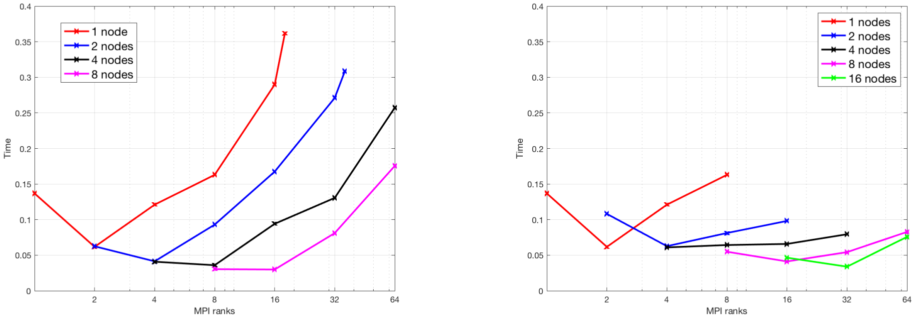
(b) Low-Synch ICWY MGS-GMRES

Figure: Performance on Eagle Skylake + V100 GPU. NREL ABL Mesh $n = 9\text{M}$

- Extended strong scaling roll-off by 4x for ExaWind ABL grid on Skylake + V100.

ECP EXASCALE COMPUTING PROJECT

Low-Synch GMRES Multi-MPI Multi-GPU



(a) Hypre Gram-Schmidt Times (b) Low-Synch Gram-Schmidt Time

Figure: Gram-Schmidt kernels time on Eagle Skylake + V100 GPU. NREL ABL Mesh $n = 9M$.

Gram-Schmidt times are flat for low-synch algorithm(s)

EXASCALE COMPUTING PROJECT

McAlister Blade

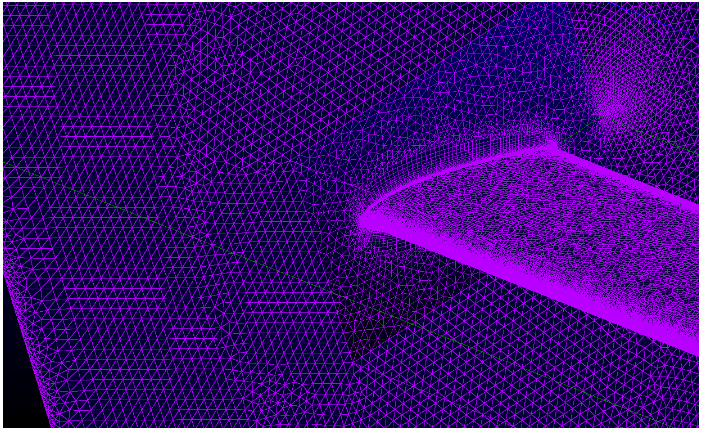


Figure: McAlister blade. Computational mesh.

McAlister Blade

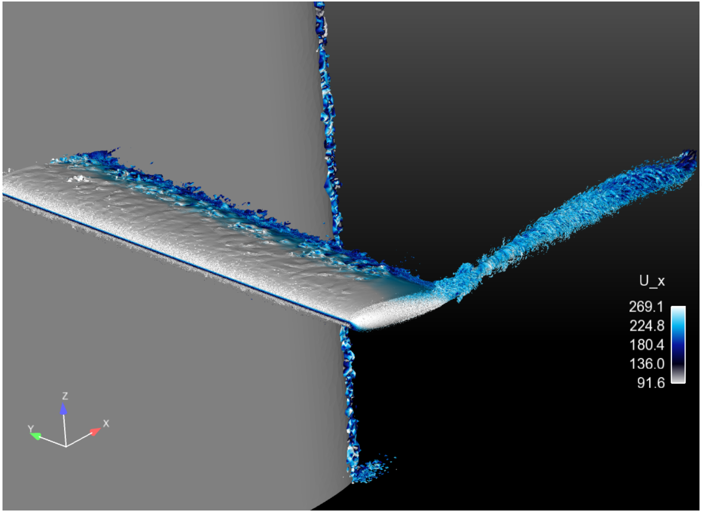


Figure: McAlister blade. Tip vortex

GMRES-AMG reduced iterations with CGS-2 projection

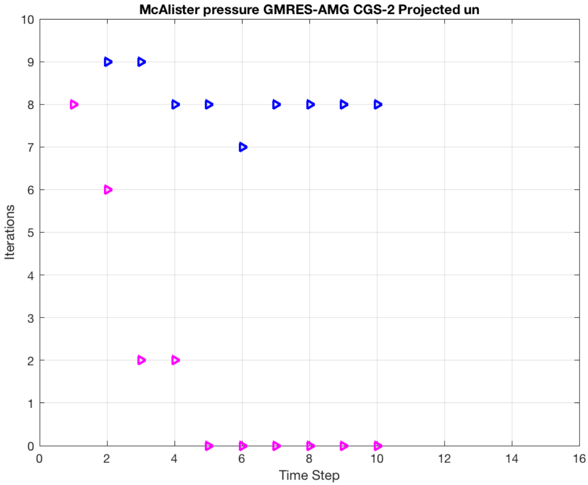


Figure: Projected x_0. CGS-2. Nalu-Wind pressure solve. McAlister blade.

Iterative Classical Gram-Schmidt CGS-2

Algorithm 3 Iterated Classical Gram-Schmidt (CGS-2) Algorithm

Input: Matrices Q_{j-1} and R_{j-1}, $A_{j-1} = Q_{j-1}R_{j-1}$; column vector $q_j = a_j$
Output: Q_j and R_j, such that $A_j = Q_jR_j$

for $k = 1, 2$ **do**
  $s^{(k-1)} = Q_{j-1}^T \, a^{(k-1)}$ ▷ Global synch
  $r_{1:j-1,j}^{(k)} = r_{1:j-1,j}^{(k)} + s^{(k-1)}$
  $q_j = q_j - Q_{j-1} \, s^{(k-1)}$
end for
$q_j = a^{(k)}/r_{jj} = \|a^{(k)}\|_2$ ▷ Global synch
$r_j = [\, r_{1:j-1,j}, \, r_{jj} \,]$

Recursive Classical Gram-Schmidt CGS-2

Orthogonal projector takes the form, $A = QR$,

$$P^{IC} = I - [\, Q_{j-2},\, q_{j-1} - Q_{j-2} Q_{j-2}^T q_{j-1} \,]\, T_{j-1}\, [\, Q_{j-2},\, q_{j-1} \,]^T$$

Correction matrix $(\, Q_{j-2}^T\, Q_{j-2} \,)^{-1} = T_{j-1} = I - L_{j-1} - L_{j-1}^T$ takes the form

$$T_{j-1} = \begin{bmatrix} T_{j-2} & T_{:,1:j-2} \\ T_{1:j-2,:} & t_{j-1,j-1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ -w^T \alpha^{-1} & 1 \end{bmatrix}$$

$w^T = L_{j-2,:}$. Thus we have, $r_{1:j-2,j} = z$, $\alpha = r_{j-1,j-1}^{-1}$ and where

$$[\; w, \quad z \;] = \left[\; Q_{j-2}^T\, q_{j-1}, \quad Q_{j-2}^T\, a_j \;\right], \quad [\; r_{j-1,j-1}, \quad r_{j-1,j} \;] = \left[\; (\, q_{j-1}^T q_{j-1} - w^T w \,), \quad (\, q_{j-1}^T\, a_j - w^T z \,) \;\right]$$

Faster solver and more accurate eigenvalue computations based on Lanczos/Arnoldi algorithms PETSc-SLEPc. Corrects the Hernandez, Roman, Tomas (2005), (2007) DCGS-2 algorithm

Rolling Stones Theorem: You can't always get what you want,
But if you try sometime you find, you get what you need

Algorithm 4 Classical Gram-Schmidt (CGS-2) Algorithm with Normalization Lag and Reorthogonalization Lag

Input: Matrices Q_{j-1} and R_{j-1}, $A_{j-1} = Q_{j-1} R_{j-1}$; column vector $q_j = a_j$
Output: Q_j and R_j, such that $A_j = Q_j R_j$

```
if j = 1 return
[r_{j-1,j-1}, r_{j-1,j}] = q_{j-1}^T [q_{j-1}, q_j]                       ▷ Global synch
if j > 2 then
    [w, z] = Q_{j-2}^T [q_{j-1}, q_j],                                      ▷ same global synch
    [r_{j-1,j-1}, r_{j-1,j}] = [r_{j-1,j-1} - w^T w, r_{j-1,j} - w^T z]     ▷ Pythagorean
    r_{1:j-2,j-1} = r_{1:j-2,j-1} + w                                       ▷ Lagged R update
end if
r_{j-1,j-1} = { r_{j-1,j-1} }^{1/2}                                         ▷ Lagged norm
if j > 2 q_{j-1} = q_{j-1} - Q_{j-2} w                                      ▷ Lagged Reorthogonalization
q_{j-1} = q_{j-1} / r_{j-1,j-1}                                             ▷ Lagged Normalization
r_{j-1,j} = r_{j-1,j} / r_{j-1,j-1}
r_{1:j-2,j} = z                                                             ▷ Apply recursive projection
q_j = q_j - Q_{j-1} r_j
```

Pythagorean form (can) mitigate cancellation. $a^2 - b^2 = (a-b)(a+b)$
Smoktunowicz, Barlow, Langou (2008). Ghysels et al (2013)

Backward (representation) error

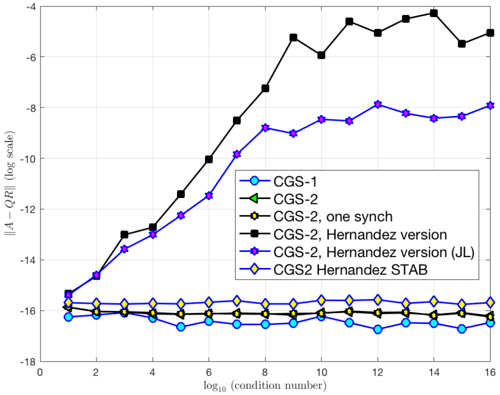


Figure: Representation error for classical Gram-Schmidt

Orthogonality

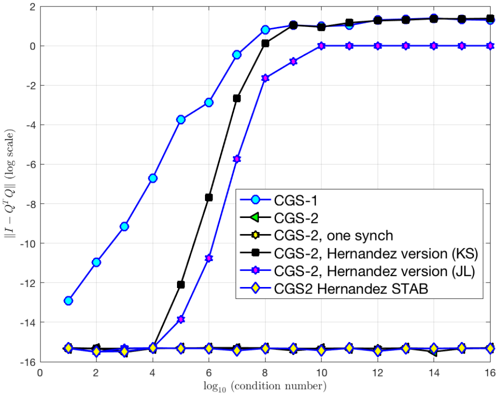


Figure: Orthogonality for classical Gram-Schmidt

EXASCALE COMPUTING PROJECT

Trilinos *s*-step GMRES

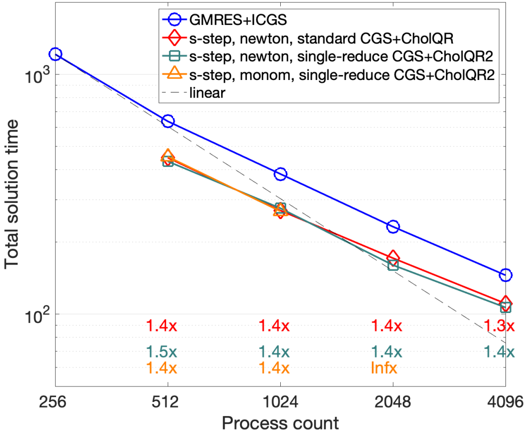


Figure: McAlister Solve Time.

Trilinos *s*-step GMRES

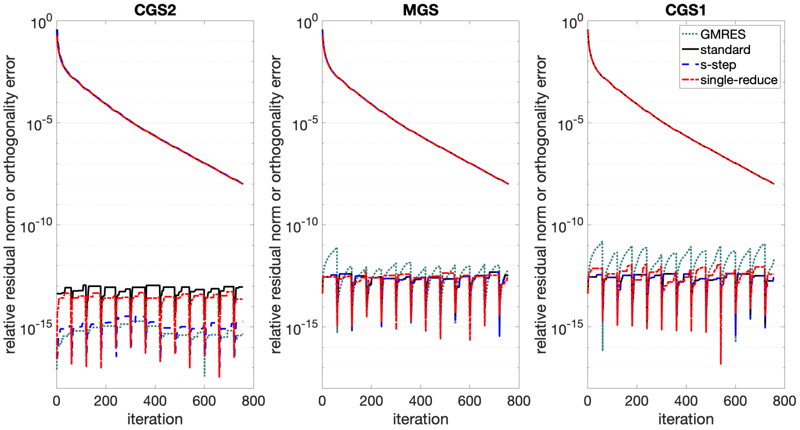


Figure: Convergence for *s*-step GMRES. Laplacian matrix

Trilinos *s*-step GMRES

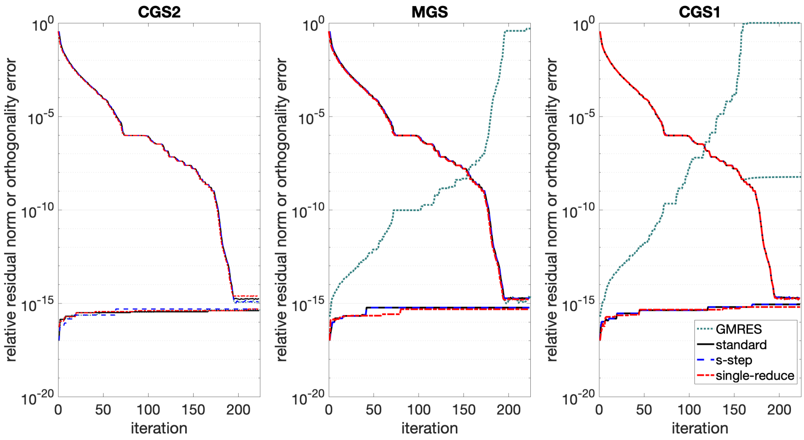


Figure: Convergence for *s*-step GMRES. Steam-1 matrix

Trilinos *s*-step GMRES

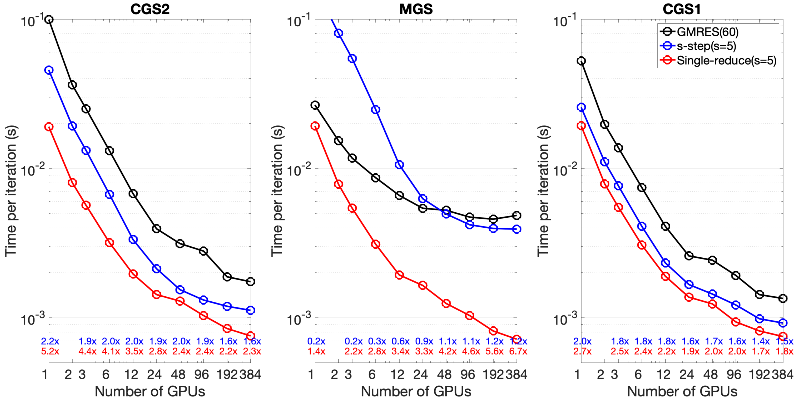


Figure: Time per iteration *s*-step GMRES.

Trilinos *s*-step GMRES

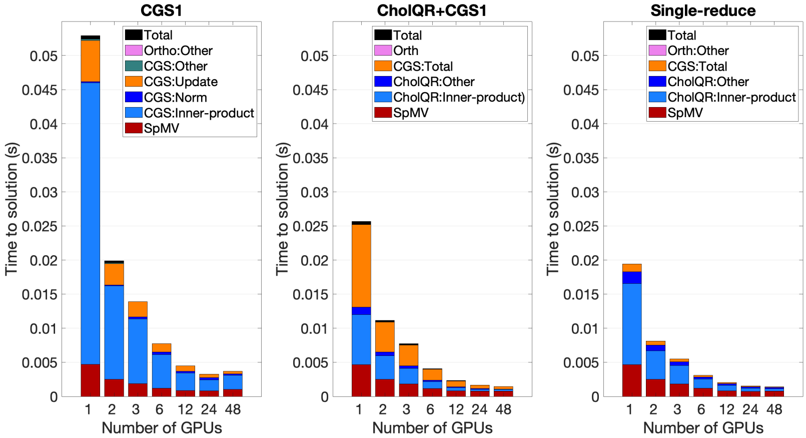


Figure: Time per iteration *s*-step GMRES.

Trilinos *s*-step GMRES

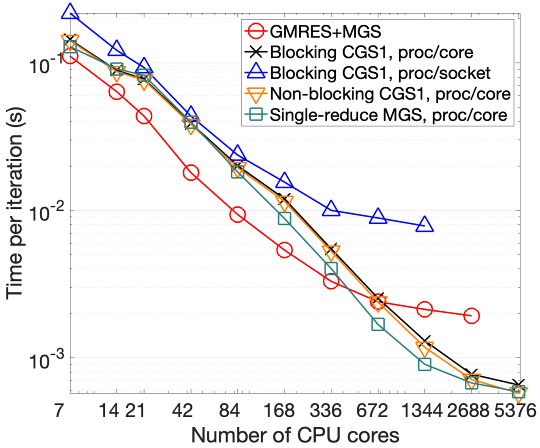


Figure: Strong scaling *s*-step GMRES versus one-sync GMRES.

EXASCALE COMPUTING PROJECT

Benzi-Tuma AINV Biconjugation

Sparse approximate inverse for non-symmetric A

$$W^T A Z = D, \quad A = W^{-T} D Z^{-1}, \quad A = LDU$$

Analogous to incomplete *ILU* factorization.

Benzi and Tuma (1998), Oblique-norm Gram-Schmidt algorithm

- L. Fox (1966), presented non-symmetric algorithm
- S. Thomas (1992) analysed the loss of orthogonality
- J. Kupal, M. Tuma, M. Rozložnik, A. Smoktunowicz (2014)
- R. Lowerey and J. Langou (2018), representation error

EXASCALE COMPUTING PROJECT

AINV Lower Triangular Smoother

Smoother applied as Richardson iteration local to MPI-rank

$$x_{k+1} = x_k + D^{-1} W^T (b - A x_k)$$

Hybrid M_H or l_1 sub-domain smoothers for Hypre-BoomerAMG

$$G = I - M_H^{-1} A, \quad M_H = \mathrm{diag}(B_k)$$

Speed advantage of matrix-vector multiplication on GPU

- Lower sweep count than Gauss-Seidel in V-cycle
- Faster GMRES-AMG convergence
- Triangular matvec is at least $25\times$ faster the cuSparse triangular solver

EXASCALE COMPUTING PROJECT

AINV Smoothers for GPU

McAlister 3M DOF. GMRES-AMG pressure solver convergence with AINV smoother

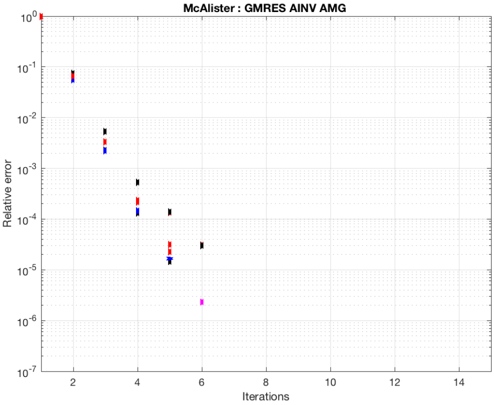


Figure: Pressure GMRES+AMG. AINV (blue 2 sweeps), Gauss-Seidel (black 2 sweeps)

Challenges to be addressed

Scientific

- AINV for Trilinos-Muelu smoothers in *s*-step CA-GMRES,
- AINV preconditioner to replace SGS in momentum

Technical

- Testing performance on Summit (multi-node)
- Verify low AMG set-up time on GPU - Trilinos-MueLu and Hypre-BoomerAMG
- Integration with the Nalu-Wind model, Trilinos-Belos-Muelu and Boomer-AMG Hypre Stacks

Thank you! Questions?

Acknowledgements:

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation's exascale computing imperative.
- This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. The NREL Eagle supercomputer was also used extensively for these studies
- NREL is a national laboratory of the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, operated by the Alliance for Sustainable Energy, LLC under Contract No. DE-AC36-08GO28308.

EXASCALE COMPUTING PROJECT