s-Step Krylov Methods as Bottom Solvers for Geometric Multigrid

Samuel Williams[1],
Erin Carson[2], Nick Knight[2], Mike Lijewski[1],
Ann Almgren[1], Brian Van Straalen[1], James Demmel[2]

[1]Lawrence Berkeley National Laboratory
[2]University of California at Berkeley

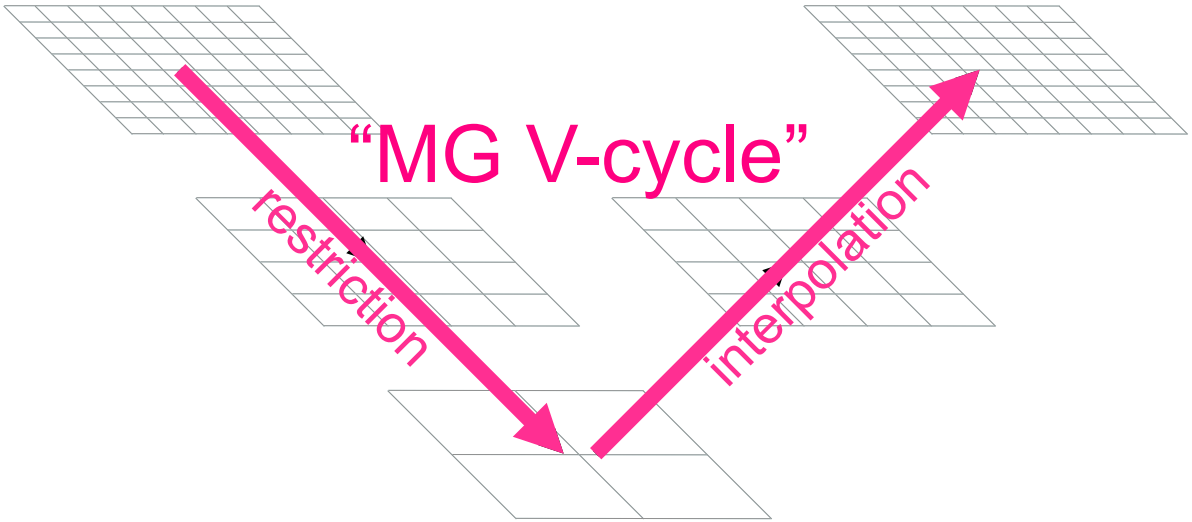
SWWilliams@lbl.gov

Outline

- Introduction to AMR MG
- Bottlenecks in MG Solvers
- s-step BiCGStab (CABiCGStab) algorithm
- CABiCGStab in miniGMG
- CABiCGStab in Real Applications
- Conclusions
- Future Work

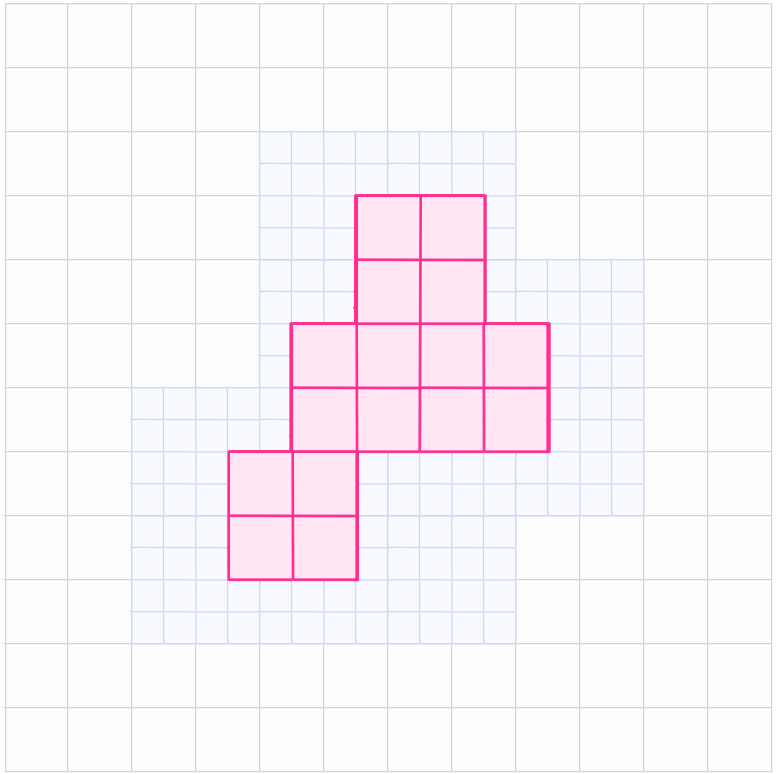
Multigrid Introduction

- Linear Solvers (Ax=b) are ubiquitous in scientific computing...
  - Combustion, Climate, Astrophysics, Cosmology, etc…
- Multigrid solves elliptic PDEs using a hierarchical approach
  - O(N) computational complexity
  - Geometric Multigrid is specialization in which the linear operator (A) is simply a stencil on a structured grid (i.e. *matrix-free*)



Multigrid in Adaptive Mesh Refinement Applications

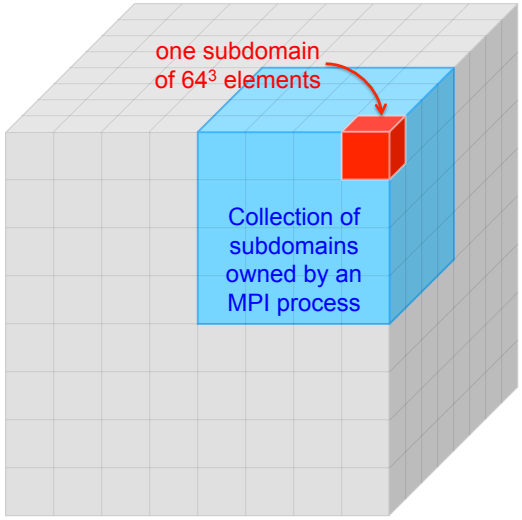
- Start with a coarse AMR level
- Add progressively finer AMR levels as needed
- In AMR applications, one performs MG solves on different AMR levels.
- Unfortunately, one can reach a point where **further geometric restriction is not possible.**
- To solve this potentially large **coarse grid ("bottom") problem**, there are a number of approaches:
  - Point Relaxation (slow)
  - Direct solver (slow)
  - Switch to Algebraic Multigrid (challenging to implement C/F BC's)
  - **Use an iterative Krylov Solver like BiCGStab (BoxLib/Chombo)**



Classical BiCGStab Performance in Geometric Multigrid

miniGMG

- **miniGMG**
  - compact 3D geometric multigrid Benchmark
  - can be used to evaluate performance bottlenecks in MG+Krylov methods and prototype new algorithms.
  - **Highly instrumented for detailed timing analysis**



- **We configured miniGMG to proxy BoxLib AMR applications…**
  - Cubical domain decomposed into **one 64^3 subdomain per processor**
  - U-cycle terminated when **subdomains are coarsened to 4^3**
  - Gauss Seidel, Red-Black (“GSRB”) smoother
  - **BiCGStab bottom solver** (matrix is never explicitly formed)

Classical BiCGStab…

Algorithm 1 Classical BiCGStab for solving $Ax = b$

1: Start with initial guess x_0
2: $p_0 := r_0 := b - Ax_0$
3: Set $\tilde{r}$ arbitrarily so that $(\tilde{r}, r_0) \neq 0$
4: **for** $j := 0, 1, \ldots$ until convergence or breakdown **do**
5: $\quad \alpha_j := (\tilde{r}, r_j)/(\tilde{r}, Ap_j)$
6: $\quad x_{j+1} := x_j + \alpha_j p_j$
7: $\quad q_j := r_j - \alpha_j Ap_j$
8: $\quad$ Check $||q_j||_2 = (q_j, q_j)^{1/2}$ for convergence
9: $\quad \omega_j := (q_j, Aq_j)/(Aq_j, Aq_j)$
10: $\quad x_{j+1} := x_{j+1} + \omega_j q_j$
11: $\quad r_{j+1} := q_j - \omega_j Aq_j$
12: $\quad$ Check $||r_{j+1}||_2 = (r_{j+1}, r_{j+1})^{1/2}$ for convergence
13: $\quad \beta_j := (\alpha_j/\omega_j)(\tilde{r}, r_{j+1})/(\tilde{r}, r_j)$
14: $\quad p_{j+1} := r_{j+1} + \beta_j(p_j - \omega_j Ap_j)$
15: **end for**

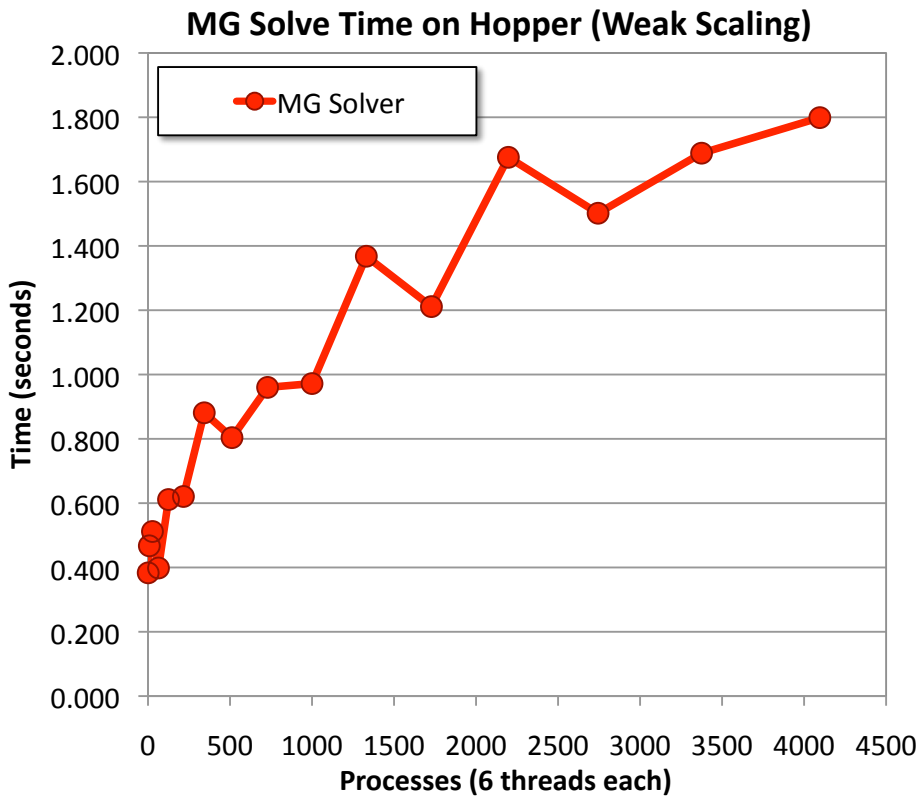
- BiCGStab solves Ax=b
  - vectors x and b are cell-centered structured grids (with ghost zones) partitioned across multiple nodes.
  - matrix A is a stencil
  - requires a few auxiliary vectors (grids)
- Observe that for each iteration, the classical BiCGStab performs…
  - 2 matvecs (Isend/Irecv)
  - 4 dot products (AllReduce)
  - 2 norms (AllReduce)

Baseline Performance

- Run miniGMG benchmark on Hopper (Cray XE6 at NERSC)
- synthetic problem:
  - variable coefficient helmholtz
  - periodic boundary conditions
  - rhs = sum of triangle waves in 3D each with one period across the entire domain
- **Weak scale to 24K cores…**
  - one 64^3 box per 6-thread process

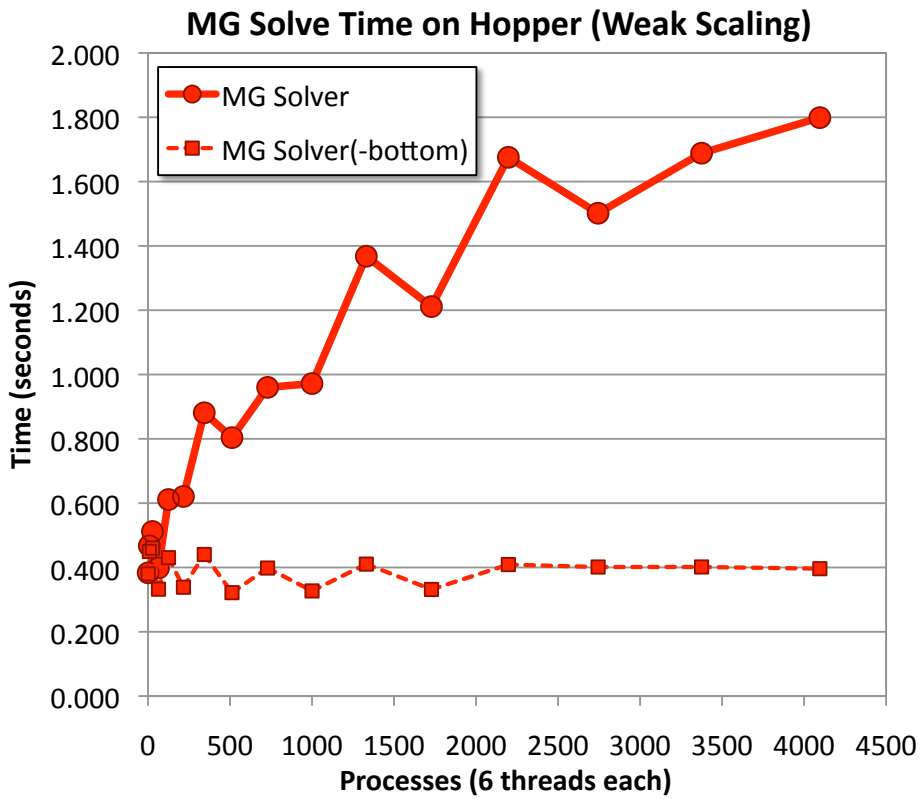
Baseline Performance

- Run miniGMG benchmark on Hopper (Cray XE6 at NERSC)
- synthetic problem:
  - variable coefficient helmholtz
  - periodic boundary conditions
  - rhs = sum of triangle waves in 3D each with one period across the entire domain
- **Weak scale to 24K cores…**
  - one 64^3 box per 6-thread process
- Although multigrid's O(N) complexity should yield constant time-to-solution when scaling, it is **clear time-to-solution is far from constant**.



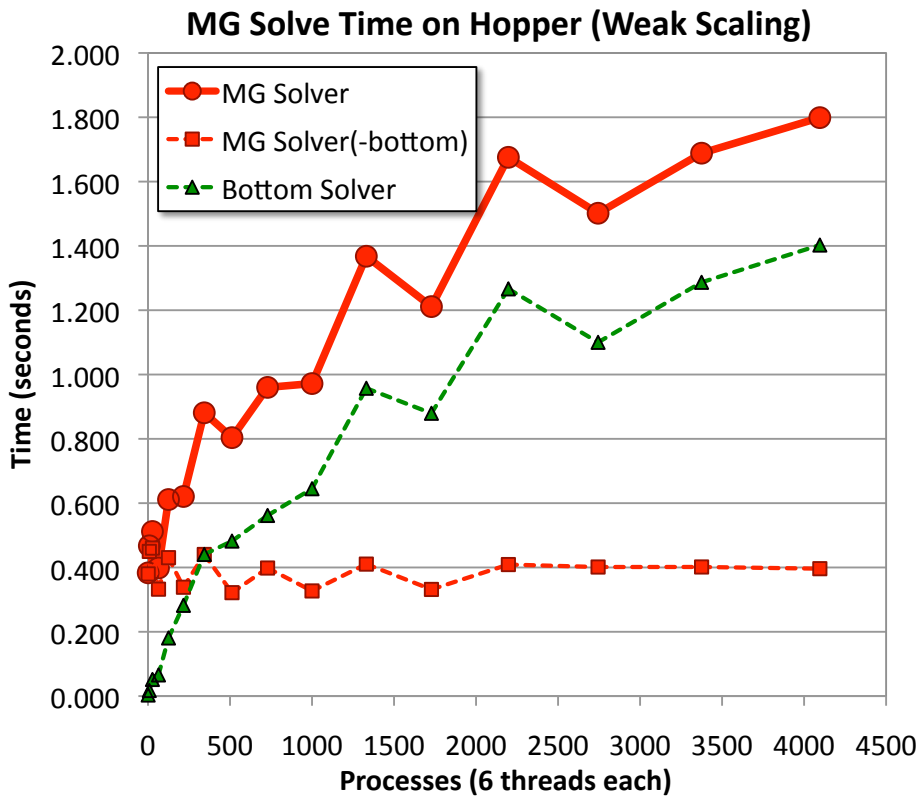
Baseline Performance

- using miniGMG's fine-grained timing instrumentation…
- time outside the bottom solver (traditional multigrid) is constant (**perfect scaling**)



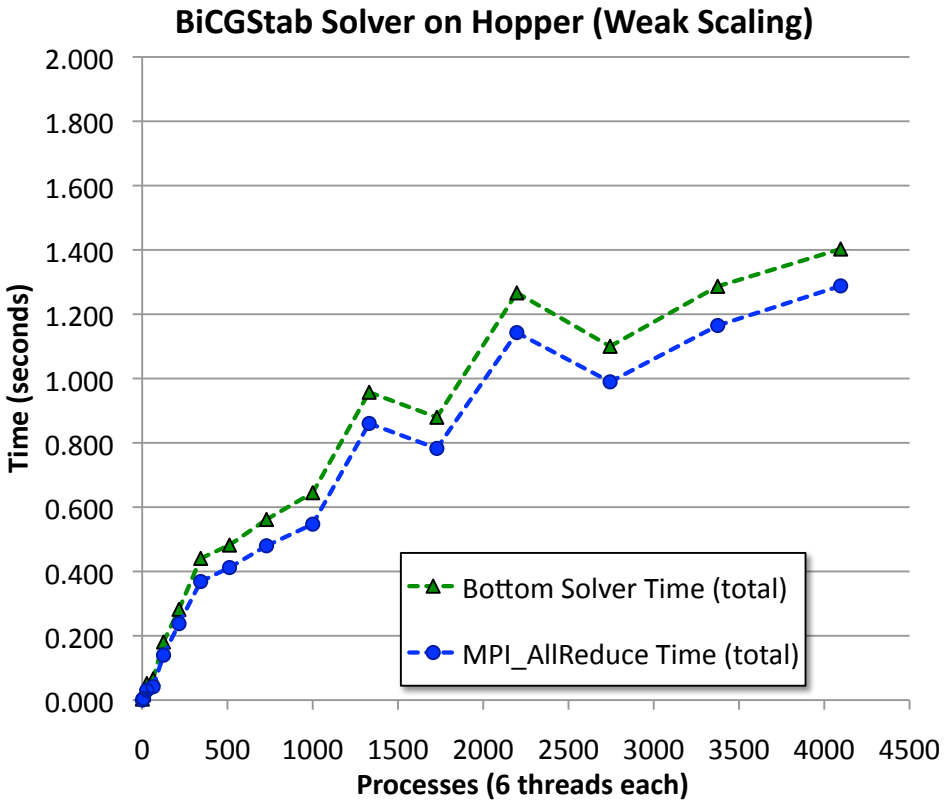
Baseline Performance

- using miniGMG's fine-grained timing instrumentation…
- time outside the bottom solver (traditional multigrid) is constant (**perfect scaling**)
- However, the time in the **bottom solver scales very poorly**



Bottom Solver Performance

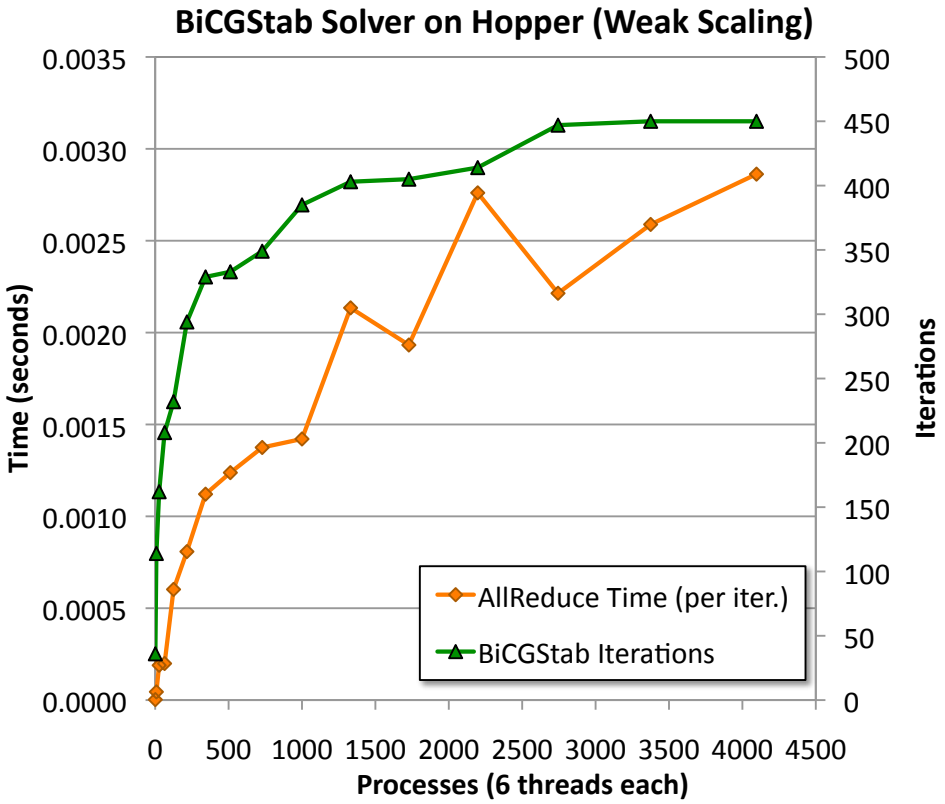
- using miniGMG's fine-grained timing instrumentation…
- time outside the bottom solver (traditional multigrid) is constant (**perfect scaling**)
- However, the time in the **bottom solver scales very poorly**
- Total time in MPI_AllReduce (used for norm's and dot's) increases rapidly with scale.



Bottom Solver Performance

- We observe that both…
  - **total number of BiCGStab iterations increases with problem size**
  - **time per MPI_AllReduce() increases with scale**
- Combined, these have a multiplicative effect…

  ever more ever slower iterations
- Four options:
  - Accelerate Collectives
  - Hide Time in Collectives
  - **Amortize Collectives**
  - Eliminate Collectives altogether



s-step BiCGStab (CABiCGStab)

Classical BiCGStab to s-Step BiCGStab (CABiCGStab)

Algorithm 1 Classical BiCGStab for solving $Ax = b$

1: Start with initial guess x_0
2: $p_0 := r_0 := b - Ax_0$
3: Set $\tilde{r}$ arbitrarily so that $(\tilde{r}, r_0) \neq 0$
4: **for** $j := 0, 1, \ldots$ until convergence or breakdown **do**
5: $\quad \alpha_j := (\tilde{r}, r_j)/(\tilde{r}, Ap_j)$
6: $\quad x_{j+1} := x_j + \alpha_j p_j$
7: $\quad q_j := r_j - \alpha_j Ap_j$
8: $\quad$ Check $||q_j||_2 = (q_j, q_j)^{1/2}$ for convergence
9: $\quad \omega_j := (q_j, Aq_j)/(Aq_j, Aq_j)$
10: $\quad x_{j+1} := x_{j+1} + \omega_j q_j$
11: $\quad r_{j+1} := q_j - \omega_j Aq_j$
12: $\quad$ Check $||r_{j+1}||_2 = (r_{j+1}, r_{j+1})^{1/2}$ for convergence
13: $\quad \beta_j := (\alpha_j/\omega_j)(\tilde{r}, r_{j+1})/(\tilde{r}, r_j)$
14: $\quad p_{j+1} := r_{j+1} + \beta_j(p_j - \omega_j Ap_j)$
15: **end for**

- Erin Carson, Nick Knight, and Jim Demmel derived s-step variants of BiCG and BiCGStab [5]…
- Blocking BiCGStab's iteration space into blocks of s-steps
- Constructing a Krylov subspace [P,R] which spans powers of A applied to p_m and r_m
- length-N vectors of BiCGStab can be expressed in terms of the product of [P,R] and length-(4s+1) vectors a,c,d,e.
- $[Aq_j, Ap_j] = [P,R]T'[d_j, a_j]$, where T' is a small locally-replicated matrix
- With a little manipulation, BiCGStab's dot products can be expressed in terms of a Gram-like matrix and one arrives at the s-step algorithm…

[5] Erin Carson, Nicholas Knight, James Demmel, "Avoiding communication in nonsymmetric Lanczos-based Krylov subspace methods", SIAM J. Sci. Comp., 2013.

CABiCGStab
(algorithm)

Algorithm 2 CABiCGStab for solving $Ax = b$

1: Start with initial guess x_0
2: $p_0 := r_0 := b - Ax_0$
3: Set $\tilde{r}$ arbitrarily so that $(\tilde{r}, r_0) \neq 0$
4: Construct $(4s+1)$-by-$(4s+1)$ matrix T'
5: **for** $m := 0, s, 2s, \ldots$ until convergence or breakdown **do**
6: Compute P, a basis for $\mathcal{K}_{2s+1}(A, p_m)$
7: Compute R, a basis for $\mathcal{K}_{2s}(A, r_m)$
8: $[G, g] := [P, R]^T [P, R, \tilde{r}]$
9: Initialize length-$(4s+1)$ vectors a_0, c_0, d_0, e_0
10: **for** $j := 0$ to $s-1$ (or convergence/breakdown) **do**
11: $\alpha_{m+j} := (g, c_j)/(g, T'a_j)$
12: $e_{j+1} := e_j + \alpha_{m+j} a_j$
13: $d_j := c_j - \alpha_{m+j} T' a_j$
14: Check $||q_{m+j}||_2 = (d_j, Gd_j)^{1/2}$ for convergence
15: $\omega_{m+j} := (d_j, GT'd_j)/(T'd_j, GT'd_j)$
16: $e_{j+1} := e_{j+1} + \omega_{m+j} d_j$
17: $c_{j+1} := d_j - \omega_{m+j} T' d_j$
18: Check $||r_{m+j+1}||_2 = (c_{j+1}, Gc_{j+1})^{1/2}$ for convergence
19: $\beta_{m+j} := (\alpha_{m+j}/\omega_{m+j})(g, c_{j+1})/(g, c_j)$
20: $a_{j+1} := c_{j+1} - \beta_{m+j}(a_j - \omega_{m+j} T' a_j)$
21: **end for**
22: $p_{m+s} := [P, R] a_s$
23: $r_{m+s} := [P, R] c_s$
24: $x_{m+s} := [P, R] e_s + x_m$
25: **end for**

- In exact arithmetic, the s-step algorithm **exactly reproduces the classical BiCGStab algorithm.**
- Computation of [P,R] can be done sequentially, in pairs, or in a communication-avoiding (minimize DRAM or #messages) manner.
- Construction of [G,g] is essentially an odd-shaped matrix multiplication, but **can be performed with only one AllReduce.**
- There is **no communication in the inner s-steps of the algorithm**.
  - operations are on the small locally-replicated vectors a, c, d, e
  - Convergence checks may be performed without additional communication.
- Updating the iterates (BiCGStab vectors) **requires no communication**.

Bottleneck Analysis Drives Implementation Choices

- We can decompose CABiCGStab's run time into three major components:

Construction of Krylov Subspace: $[P,R] = [p,Ap,\ldots,A^{2s}p,\ r,Ar,\ldots,A^{2s-1}r]$

Gram-like Matrix: $[G,g] = [P,R]^T[P,R,rt]$

P2P MPI + local matvecs + **global MPI collectives**

- In multigrid's coarse grid solve, local matvecs are free, and MPI collectives dominate run time.
- We **implemented CABiCGStab in miniGMG**…
  - Construct [P,R] sequentially (not performance critical)
  - **Optimized construction of [G,g] to use only 1 collective.**

Bottleneck Analysis Drives Implementation Choices

- Other users of BiCGStab (those not using MG) might see the matvecs dominate the run time.

Construction of Krylov Subspace:
$[P,R] = [p,Ap,\ldots,A^{2s}p,\ r,Ar,\ldots,A^{2s-1}r]$

Gram-like Matrix:
$[G,g] = [P,R]^T[P,R,rt]$

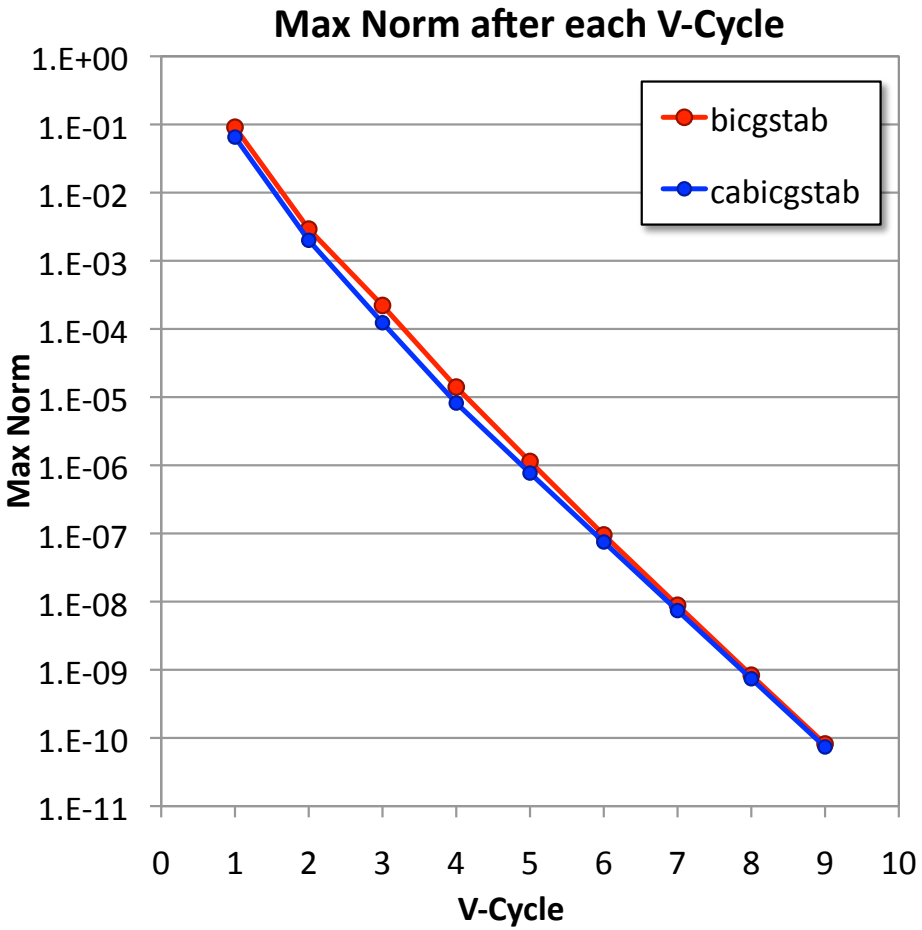
P2P MPI + **local matvecs** + global MPI collectives

- They should optimize the implementation of the s-step BiCGStab algorithm aifferently…
  - minimize vertical (DRAM) data movement
  - calculate $[Ap,A^2p,Ar,A^2r]$ by reading A only once

CABiCGStab in miniGMG

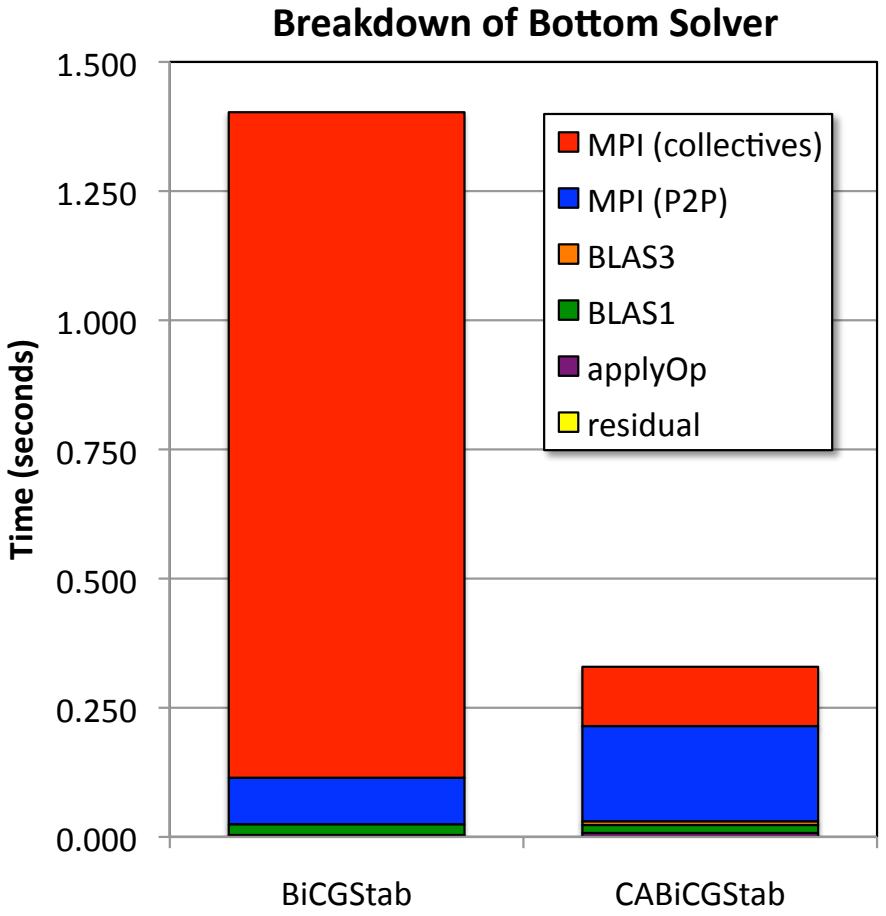
CABiCGStab
(accuracy)

- miniGMG with CABiCGStab (s=4) has **the same convergence rate** as using BiCGStab
- Note, CABiCGStab uses the L2 norm for convergence



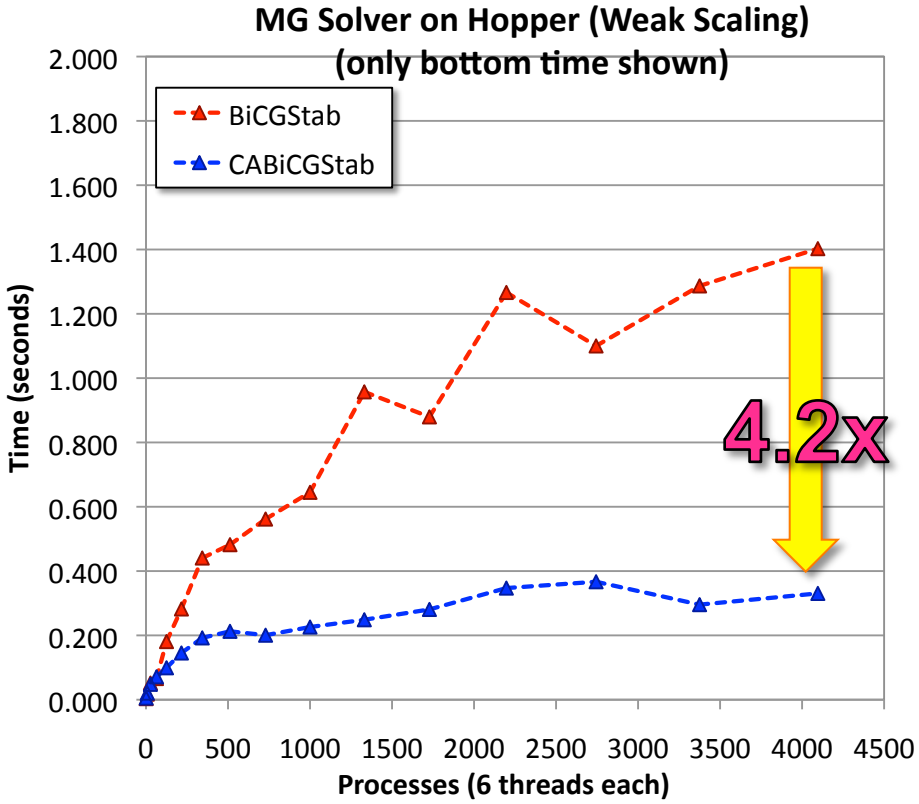
CABiCGStab
(breakdown of bottom solve time)

- CABiCGStab **replaces 6s scalar reductions with one matrix reduction**.
- CABiCGStab **requires twice the peer-to-peer MPI communication** per s steps as the classical algorithm.
- We observe reduction in collective time outweighed increase in P2P time.



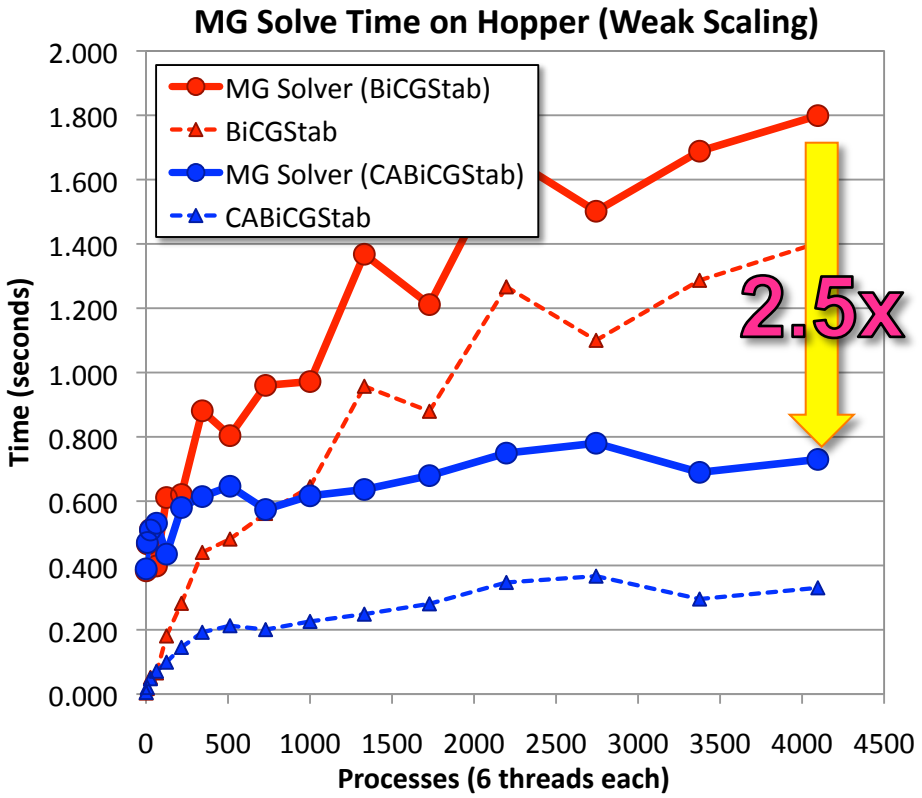
Benefits to miniGMG from CABiCGStab

- Replaced BiCGStab bottom solver in miniGMG with CABiCGStab and ran scaling experiments…
- At 4K processes, CABiCGStab more than **quadrupled the bottom solver performance**.



Benefits to miniGMG from CABiCGStab

- Replaced BiCGStab bottom solver in miniGMG with CABiCGStab and ran scaling experiments…
- At 4K processes, CABiCGStab more than **quadrupled the bottom solver performance**.
- Moreover, it provided MG with a **2.5x overall speedup**.
- Thus, it dramatically improved parallel efficiency.



Can CABiCGStab help Real Applications?

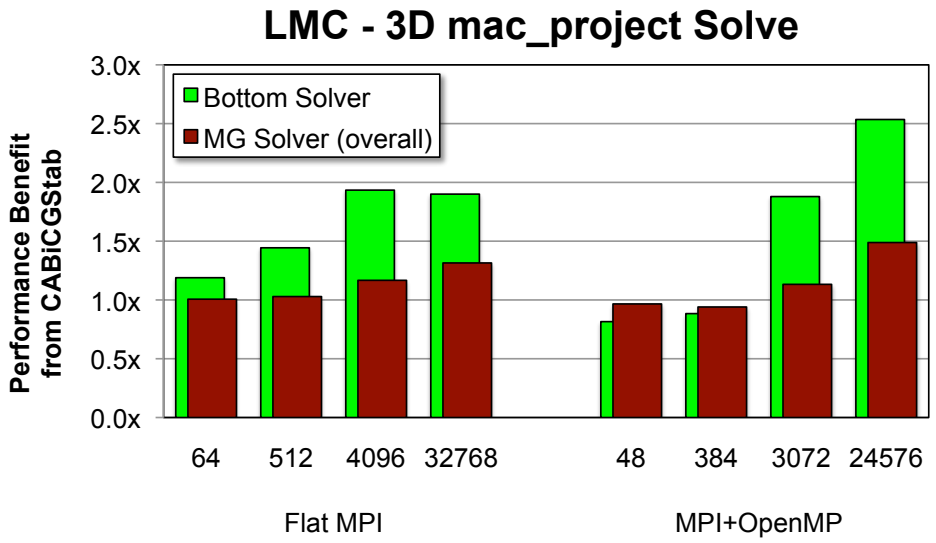
BoxLib MG Solvers

- BoxLib is an AMR MG framework developed at LBL.
  - uses BiCGStab as a coarse-grid solver
  - includes both C++ and Fortran versions of BiCGStab.

- We implemented both C++ and Fortran versions of CABiCGStab…
  - **allows drop in replacement for BiCGStab**
  - exploits all existing infrastructure for applyOp, BCs, ghost zones, etc…
  - allows for rapid evaluation on real applications
  - We exploited BoxLib capabilities to construct [P,R] in pairs.
    *this keeps the number of **messages equal to the classical version**.*
  - We implemented a **Telescoping CABiCGStab algorithm** in which we steadily increase s.

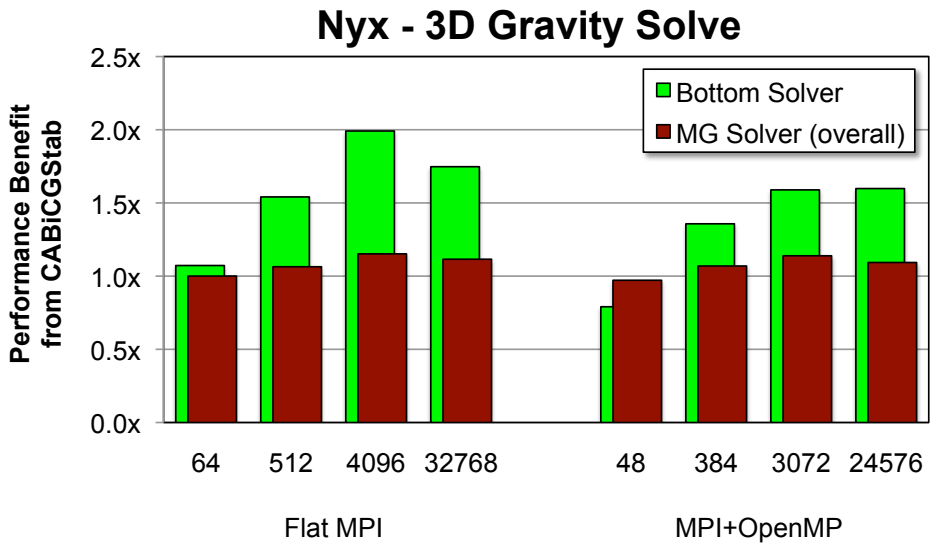
LMC Combustion Application

- **Low Mach Number Adaptive Mesh Refinement Code (LMC)**
  - Navier-Stokes
  - reactive chemistry
  - AMR
- MG Diffusion Solve
  - $(a\alpha - b\nabla\beta\nabla)u = f$
  - require relatively few bottom solver iterations to converge
  - CABICGStab not applicable
- AMR MG Level Solve
  - $b\nabla\beta\nabla u = f$
  - **require lots of bottom solve iterations to convergence**
- Conducted scaling experiments to 32K cores on Hopper (XE6 at NERSC)
- Benefit of CABiCGStab in 3D:
  - **up to 2.5x for the bottom solve**
  - **up to 1.5x overall for the MG level solve**



Nyx Cosmology Application

- **Cosmological dark matter simulation code (SciDAC)**
- Code is a mix of:
  - hydrodynamics for gas
  - cloud-in-cell particles for dark matter
- Poisson solve for gravitational potential…
  - multi-level AMR MG
  - constant coefficient
  - $b\nabla^2 u = f$



- Conducted scaling experiments to 32K cores on Hopper
- Benefit of CABiCGStab
  - **up to 2x win in bottom solve**
  - Unfortunately, bottom solver was only 26-41% of the solve time.
  - **less than 15% speedup overall**

Conclusions

Conclusions

- Geometric multigrid solvers can be bottlenecked by the performance and scalablilty of their coarse-grid (bottom) solvers…
  - Degraded MPI collective performance
  - super linear computational complexity of Krylov methods

- Communication-Avoiding s-step methods:
  - provide a **drop in-replacement for BiCGStab**
  - asymptotically **reduce the number of collective** operations
  - are ultimately **bounded by P2P MPI communication**.
  - yeild significant speedups on both synthetic and real-world AMR MG solves

- CA Krylov methods provide an interesting axis for co-design research:
  - **trade latency (collectives/P2P) performance for bandwidth**
  - trade O(s) fine-grained operations for one coarse-grained operation
  - trade streaming kernels for 2.5D kernels (good for locality)

Future Work

- Improve the performance of s-step bottom solvers…
  - We don't exploit the fact that the matrix G is symmetric.
  - Thus, we send twice the data we need to.
  - Potential performance impediment for large s as G has $O(s^2)$ elements.

- Explore using s-step methods for DRAM communication avoiding…
  - Large matrices/vectors don't fit in cache
  - matvec's can dominate the run time
  - Optimize CABiCGStab (stencil powers or matrix powers) for DRAM data movement as previous efforts optimized CAGMRES

- Explore true distributed v-cycles in AMR MG solves…
  - Eliminates collectives altogether
  - Geometric approach requires integrating the complex BC's endemic to AMR into the restriction operations
  - Algebraic approach must express the BC's inside an explicit matrix.
  - As AMG is memory hungry, it probably should only be applied to mac_project.

Acknowledgements

- All authors from Lawrence Berkeley National Laboratory were supported by the DOE Office of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231.
- This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Questions?

Backup Slides