Efficiency through reuse in algebraic multigrid

Andrey Prokopenko
Paul Lin
John Shadid
Jonathan Hu

CCR, Sandia National Labs

April 13, 2016
Motivation

- Multigrid setup times growing with number of MPI ranks
- Many simulations only change the values of a matrix, not the structure
- Workaround for scalability issues
Motivation

- Multigrid setup times growing with number of MPI ranks
- Many simulations only change the values of a matrix, not the structure
- Workaround for scalability issues

Few years ago...

<table>
<thead>
<tr>
<th>MPI ranks</th>
<th>DOFs</th>
<th>AMG setup time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>0.8M</td>
<td>9.5</td>
</tr>
<tr>
<td>1024</td>
<td>6.5M</td>
<td>10.8</td>
</tr>
<tr>
<td>8192</td>
<td>51.0M</td>
<td>12.1</td>
</tr>
<tr>
<td>65536</td>
<td>401M</td>
<td>25.5</td>
</tr>
<tr>
<td>524288</td>
<td>3.2B</td>
<td>1312</td>
</tr>
</tbody>
</table>
Main idea
Capture errors at multiple resolutions.
Algebraic Multigrid (AMG)

Two main components

- **Smoothers**
  - Approximate solve on each level
  - “Cheap” reduction of oscillatory error (high energy)
  - $S_L \approx A_L^{-1}$ on the coarsest level $L$

Main idea

Capture errors at multiple resolutions.
Algebraic Multigrid (AMG)

Two main components

- **Smoothers**
  - Approximate solve on each level
  - "Cheap" reduction of oscillatory error (high energy)
  - \( S_L \approx A_L^{-1} \) on the coarsest level \( L \)

- **Grid transfers (prolongators and restrictors)**
  - Data movement between levels
  - Reduction of smooth error (low energy)

Main idea
Capture errors at multiple resolutions.
MueLu: Trilinos multigrid library

- Can use either **EPETRA** (32-bit) or **TPETRA**
  
  **Template types:** Local and global indices, scalar, compute node

- **Grid transfers**
  - Smoothed and unsmoothed aggregation
  - Petrov-Galerkin
  - Energy minimization
  - Maxwell

- **Smoothers (IFPACK/IFPACK2)**
  - Relaxation: Jacobi, SOR, $l_1$ Gauss-Seidel
  - Incomplete factorizations: ILU(k), ILUT, ILUTP*
  - Others: Chebyshev, additive Schwarz, Krylov, Vanka, . . .

- **Direct solvers (AMESOS/AMESOS2)**
  - KLU2, SuperLU, . . .

- **Load balancing (ZOLTAN/ZOLTAN2)**
  - RCB, multijagged (ZOLTAN2 only)
Smoothed Aggregation (SA) main kernels

- Aggregation (forming coarse unknowns)
- Tentative prolongator construction $P^{\text{tent}}_\ell$
- Smoothed prolongator construction
  \[ P_\ell = (I - \omega D^{-1} A_{\ell-1}) P^{\text{tent}}_\ell \]
- Coarse matrix construction (matrix-matrix multiply)
  \[ A_\ell = R_\ell A_{\ell-1} P_\ell \]
- Load balancing of $A_\ell$ (if necessary)
- Smoother initialization
Reuse: fine level smoothers (S)

- **Reuse**
  - Symbolic factorization of smoother $S_0$
- **Recompute**
  - Smoother $S_0$ (only numeric factorization)
  - Everything else...

✓ Useful for heavy smoothers, like ILU
✓ Does not affect convergence
✗ Little benefit for light-weight smoothers
Reuse: fine level smoothers (S)

- Additive Schwarz/subdomain ILU
  - Data import infrastructure
  - Local symbolic factorizations
  - Data transfers unavoidable
- Polynomial smoothers
  - Reuse eigenvalue estimate
  - Reuse initial guess for eigenvalue estimate
- Jacobi, Gauss-Seidel, etc.
  - No reuse
Reuse: tentative prolongators (TP)

- **Reuse**
  - Tentative prolongators $P_{\ell}^{(tent)}$, $\ell > 0$
  - Matrix graphs of $P_{\ell}$ and $A_{\ell}$, $\ell > 0$
  - Symbolic factorization of smoothers $S_{\ell}$, $\ell \geq 0$

- **Recompute**
  - Smoothed prolongators $P_{\ell}$, $\ell > 0$
    (reusing matrix graphs)
  - Coarse level matrices $A_{\ell}$, $\ell > 0$
    (reusing matrix graphs)
  - Smoothers $S_{\ell}$ (only numeric factorization)

✓ Avoids construction of tentative prolongator
✓ Preserves import objects for rebalancing
✓ Does not affect convergence
✗ Requires matrix-matrix product for P and RAP
Reuse: final prolongators (RP)

- **Reuse**
  - Prolongators/restrictors $P_\ell, R_\ell, \ell > 0$
  - Matrix graphs of $A_\ell, \ell > 0$
  - Symbolic factorization of smoothers $S_\ell, \ell \geq 0$

- **Recompute**
  - Coarse level matrices $A_\ell, \ell > 0$ (reusing matrix graphs)
  - Smoothers $S_\ell$ (only numeric factorization)

- ✓ Avoids matrix-matrix product for final $P_\ell$
- ✓ Preserves import object for rebalancing $A_\ell, \ell > 0$
- ✗ May negatively affect convergence
- ✗ Requires matrix-matrix product for RAP

---

Andrey Prokopenko
Multigrid reuse
SIAM PP 2016 9 / 16
Reuse: all but fine level smoother (RAP)

- **Reuse**
  - Prolongators/restrictors $P_\ell, R_\ell, \ell > 0$
  - Coarse level matrices $A_\ell, \ell > 0$
  - Smoothers $S_\ell, \ell > 0$
  - Symbolic factorization of smoothers $S_0$

- **Recompute**
  - Smoother $S_0$ (only numeric factorization)

- ✓ No matrix-matrix products required
- ✓ Preserves coarse smoother data
- ✓ Preserves rebalancing information
- ✓ Cheapest reuse option
- ✗ Least likely to converge
Experiments: ISMIP-HOM Test C

A “first order” approximation to a full Stokes model for a standard ice sheet model benchmark ISMIP-HOM (Test C). This model is an approximation to viscous incompressible quasi-static Stokes flow with power-law viscosity

\[-\nabla \cdot (2\mu \dot{\varepsilon}_1) = -\rho g \frac{\partial s}{\partial x}\]
\[-\nabla \cdot (2\mu \dot{\varepsilon}_2) = -\rho g \frac{\partial s}{\partial y}\]

where

\[\dot{\varepsilon}_T^1 = (2\dot{\varepsilon}_{11} + \dot{\varepsilon}_{22}, \dot{\varepsilon}_{12}, \dot{\varepsilon}_{13})\]
\[\dot{\varepsilon}_T^2 = (\dot{\varepsilon}_{12}, \dot{\varepsilon}_{11} + 2\dot{\varepsilon}_{22}, \dot{\varepsilon}_{23})\]
\[\dot{\varepsilon}_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)\]

and viscosity \(\mu\) is a nonlinear function given by “Glen’s law”

\[\mu = \frac{1}{2} A^{-\frac{1}{n}} \left( \frac{1}{2} \sum_{ij} \dot{\varepsilon}_{ij}^2 \right)^{\frac{1}{2n} - \frac{1}{2}}\]

The model is complimented by relevant stress-free and floating ice boundary conditions.
Experiments: ISMIP-HOM Test C

<table>
<thead>
<tr>
<th>Reuse</th>
<th>Setup</th>
<th>Solve</th>
</tr>
</thead>
<tbody>
<tr>
<td>No reuse</td>
<td>77.2</td>
<td>66.7</td>
</tr>
<tr>
<td>TP</td>
<td>64.8</td>
<td>67.4</td>
</tr>
<tr>
<td>RP</td>
<td>42.6</td>
<td>67.5</td>
</tr>
<tr>
<td>RAP</td>
<td>22.0</td>
<td>84.8</td>
</tr>
</tbody>
</table>
Experiments: 3D MHD generator

- Steady-state 3D MHD generator
  - Resistive MHD model
  - Stabilized FE
  - Newton-Krylov solve
  - 8 DOFs/mesh node
- Monolithic preconditioner
- Prolongator is unsmoothed
- Heavy smoother
  - In this case, DD/ILU(0)
- $P$ cheap to construct compared to smoothers
Experiments: 3D MHD generator

- Steady-state 3D MHD generator
  - Resistive MHD model
  - Stabilized FE
  - Newton-Krylov solve
  - 8 DOFs/mesh node

- Monolithic preconditioner
  - Prolongator is unsmoothed
  - Heavy smoother
    - In this case, DD/ILU(0)

- $P$ cheap to construct compared to smoothers

<table>
<thead>
<tr>
<th>Reuse</th>
<th>Setup</th>
<th>Solve</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No reuse</td>
<td>3.79</td>
<td>2.65</td>
<td>6.44</td>
</tr>
<tr>
<td>S</td>
<td>3.27</td>
<td>2.63</td>
<td>5.91</td>
</tr>
<tr>
<td>RP</td>
<td>3.14</td>
<td>2.61</td>
<td>5.75</td>
</tr>
<tr>
<td>RAP</td>
<td>2.74</td>
<td>42.80</td>
<td>45.53</td>
</tr>
</tbody>
</table>
Experiments: jet problem

- 3D Jet, Re=106, CFL 0.25, no slip BCs
- SA AMG, V(3,3) symmetric Gauss-Seidel smoothing

Setup cost almost entirely
- Smoothed prolongator
  \[ P = (I - \omega D^{-1} A)P^{(tent)} \]
- Galerkin product

For this particular problem, convergence maintained
Experiments: jet problem

- 3D Jet, Re=106, CFL 0.25, no slip BCs
- SA AMG, $V(3,3)$ symmetric Gauss-Seidel smoothing
- Setup cost almost entirely
  - Smoothed prolongator
  - $P = (I - \omega D^{-1} A)P^{(tent)}$
  - Galerkin product

![Graph showing speedup with different node counts](graph.png)
Conclusions

- Easy way of reducing multigrid setup times without changing algorithms
- Multiple opportunities for reducing cost through reuse
  - Grid transfers
  - Heavy weight smoothers
  - Matrix-matrix multiplication
- Effectiveness of reusing setup information is problem dependent
- Still have some low-hanging fruit to pick
Future work

- Storing more auxiliary data
  - Temporary communication data structures
  - Hash tables for matrix-matrix multiplication
- Experimenting with reuse with next-gen smoothers
  - Multi-colored Gauss-Seidel
  - Iterative ILU
- Ability to use different reuse strategies in Newton solver and across transient steps
  - Heavy reuse within Newton solver, lighter reuse across time steps
  - Lighter reuse early in simulation due to startup conditions
- Better information exchange between nonlinear and linear solve
MueLu: references

