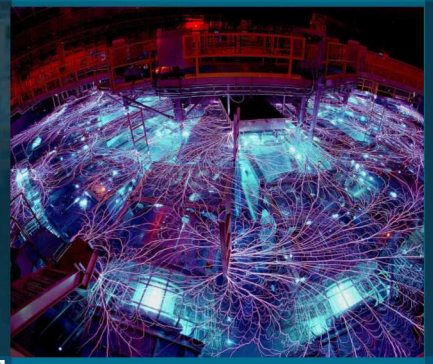
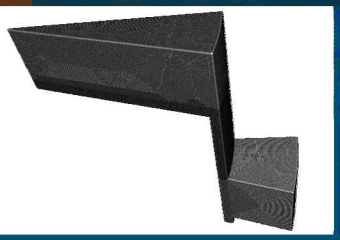
SAND2020-1793C

Linear Solvers for Plasma Simulations on Advanced Architectures





TRILINOS

Jonathan Hu, Christian Glusa, Stan Moore, Paul Lin, Edward Phillips, Matt Bettencourt, James Elliott, Chris Siefert, Siva Rajamanickam

SIAM Conference on Parallel Processing, February 12-15, 2020, SAND2020-XXXX

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline

- Motivating application
- Governing equations
- Discretization
- Algebraic multigrid solver
- Numerical results
- Future work

Motivation

Sandia mission apps are being readied for current and emerging pre-exascale architectures, as well as future exascale architectures

- DOE ASC ATDM program
- DOE Exascale Computing Project (ECP)

EMPIRE: Sandia plasma simulation code

- Relies heavily on capabilities in Sandia's Trilinos project
  - High-performance, portable shared memory primitives
  - Sparse distributed linear algebra
  - Distributed and shared memory solvers
  - Load-balancing, time-stepping
- Requires specialized algebraic multigrid (AMG) for Maxwell's equations

EMPIRE simulations have stringent solver performance requirements (10 solves/second)

EMPIRE – Sandia plasma simulation application

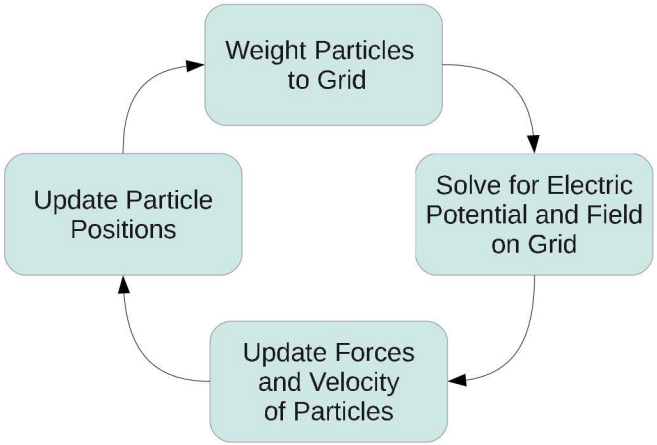
Plasma dynamics are described by Klimontovich equation

$$\frac{\partial f_i}{\partial t} + \frac{\vec{v}_i}{m} \cdot \nabla f_i + \frac{q_i}{m_i} \left(\vec{E}(x_i) + v_i \times \vec{B}(x_i) \right) \frac{\partial f_i}{\partial \vec{v}} = 0$$

for particles i with associated charge q, mass m, velocity v_i, and distribution f_i.

Particle movement coupled to electric field E and magnetic field B via Maxwell's equations.

EMPIRE uses operator-split time integration.



Bettencourt et al., *ASC ATDM Level 2 Milestone #6358: Assess Status of Next Generation Components and Physics Models in EMPIRE*, SAND2018-10100

Maxwell's Equations

$$\frac{\partial \vec{B}}{\partial t} = -\nabla \times \vec{E} \qquad \nabla \cdot (\epsilon \vec{E}) = \rho$$

$$\frac{\partial (\epsilon \vec{E})}{\partial t} = \nabla \times \mu^{-1} \vec{B} - \vec{J} \qquad \nabla \cdot \vec{B} = 0$$

$$\vec{n} \times \vec{E} = 0 \text{ on } \partial\Omega$$

electric permittivity ϵ and magnetic permeability μ.

Discretization of Maxwell's Equations

Discretized using nodal elements for $H^1(\Omega)$, Nedelec edge elements for $H(\text{curl}, \Omega)$, and Nedelec face elements for $H(\text{div}, \Omega)$.

Yields block system

$$\begin{pmatrix} \frac{1}{\Delta t}\boldsymbol{M}_B(1) & \boldsymbol{M}_B(1)\boldsymbol{C} \\ -\boldsymbol{C}^T\boldsymbol{M}_B(\mu^{-1}) & \frac{1}{\Delta t}\boldsymbol{M}_E(\varepsilon) \end{pmatrix}$$

where M_E and M_B are edge and face mass matrices, respectively

C is strong form of curl

Discretization of Maxwell's Equations (continued)

Block system can be factored

$$\begin{pmatrix} \frac{1}{\Delta t}\boldsymbol{M}_B(1) & \boldsymbol{M}_B(1)\boldsymbol{C} \\ -\boldsymbol{C}^T\boldsymbol{M}_B(\mu^{-1}) & \frac{1}{\Delta t}\boldsymbol{M}_E(\varepsilon) \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta t}\boldsymbol{M}_B(1) & \boldsymbol{0} \\ -\boldsymbol{C}^T\boldsymbol{M}_B(\mu^{-1}) & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \Delta t\boldsymbol{C} \\ \boldsymbol{0} & \boldsymbol{S}_E \end{pmatrix}$$

where the Schur complement $\boldsymbol{S}_E$ is given by

$$\boldsymbol{S}_E = \frac{1}{\Delta t}\boldsymbol{M}_E(\varepsilon) + \Delta t\boldsymbol{C}^T\boldsymbol{M}_B(\mu^{-1})\boldsymbol{C} \qquad (1)$$

Challenge: $\boldsymbol{S}_E$ has large near nullspace (space of gradients)

Linear Solver Algorithm (1)

Augmenting $\boldsymbol{S_E}$ with grad-div term to reduce nullspace:

$$\overline{\boldsymbol{S}}_{\boldsymbol{E}} = \frac{1}{\Delta t}\boldsymbol{M}_{\boldsymbol{E}}(\varepsilon) + \Delta t\left\{\boldsymbol{C}^{\boldsymbol{T}}\boldsymbol{M}_{\boldsymbol{B}}(\mu^{-1})\boldsymbol{C} + \boldsymbol{M}_{\boldsymbol{E}}(1)\boldsymbol{G}\boldsymbol{M}_{\rho}(\mu)^{-1}\boldsymbol{G}^{\boldsymbol{T}}\boldsymbol{M}_{\boldsymbol{E}}(1)\right\}$$

Solving (1) is equivalent to solving:

$$\begin{pmatrix} \overline{\boldsymbol{S}}_{\boldsymbol{E}} & \frac{1}{\Delta t}\boldsymbol{M}_{\boldsymbol{E}}(\varepsilon)\boldsymbol{G} \\ \frac{1}{\Delta t}\boldsymbol{G}^{\boldsymbol{T}}\boldsymbol{M}_{\boldsymbol{E}}(\varepsilon) & \frac{1}{\Delta t}\boldsymbol{G}^{\boldsymbol{T}}\boldsymbol{M}_{\boldsymbol{E}}(\varepsilon)\boldsymbol{G} \end{pmatrix} \begin{pmatrix} \tilde{\vec{c}}_{\boldsymbol{E}} \\ \tilde{\vec{c}}_{\rho} \end{pmatrix} = \begin{pmatrix} \vec{F} \\ \boldsymbol{G}^{\boldsymbol{T}}\vec{F} \end{pmatrix} \qquad (2)$$

where

$$\boldsymbol{S}_{\boldsymbol{E}}\vec{\boldsymbol{c}}_{\boldsymbol{E}} = \vec{F}$$

$$\vec{E} = \tilde{\vec{c}}_{\boldsymbol{E}} + \boldsymbol{G}\tilde{\vec{c}}_{\rho}$$

$$\tilde{\vec{c}}_{\boldsymbol{E}} = \boldsymbol{M}_{\boldsymbol{E}}(1)^{-1}\boldsymbol{C}\boldsymbol{M}_{\boldsymbol{B}}(\mu^{-1})\tilde{\vec{c}}_{\boldsymbol{B}}$$

Bochev et al., *An algebraic multigrid approach based on a compatible gauge reformulation of Maxwell's equations*, 2008.

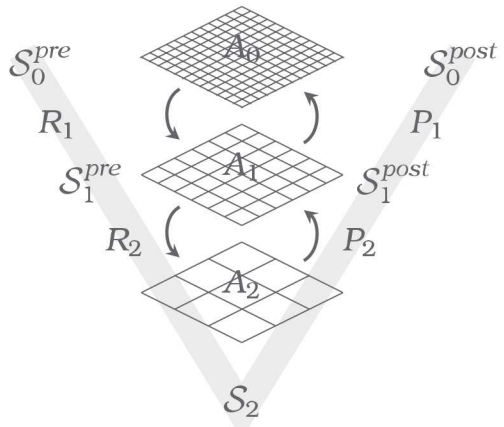
Linear Solver Algorithm (2)

Equation (2) can be solved with CG and Maxwell-specific algebraic multigrid preconditioner

- Block diagonal preconditioner
  - Off-diagonal coupling is ignored
- (1,1) block is edge-based vector Laplacian, requires special prolongator
- (2,2) block is node-based Laplacian

Multigrid: scalable solution method for linear systems arising from elliptic PDEs

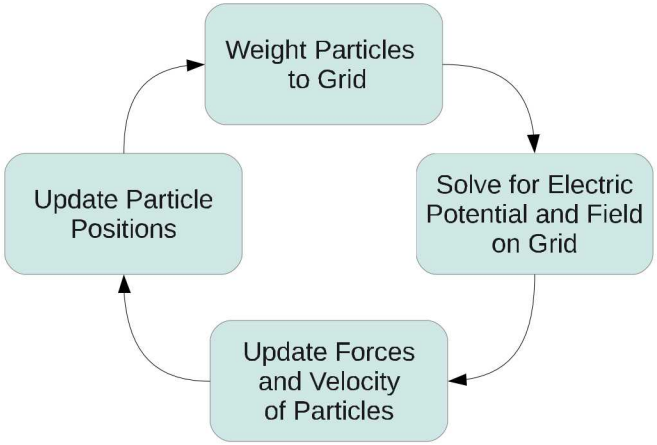
- Idea: capture error at multiple resolutions:
  - **Smoothing** reduces oscillatory error (high energy)
  - **Coarse grid correction** reduces smooth error (low energy)
- Algebraic multigrid (AMG)
- Preconditioner generates A_i's, R_i's, P_i's



Bochev et al., *An algebraic multigrid approach based on a compatible gauge reformulation of Maxwell's equations*, 2008.

Overall EMPIRE Solve

Recall that the Maxwell solve is part of overall EMPIRE solution process;

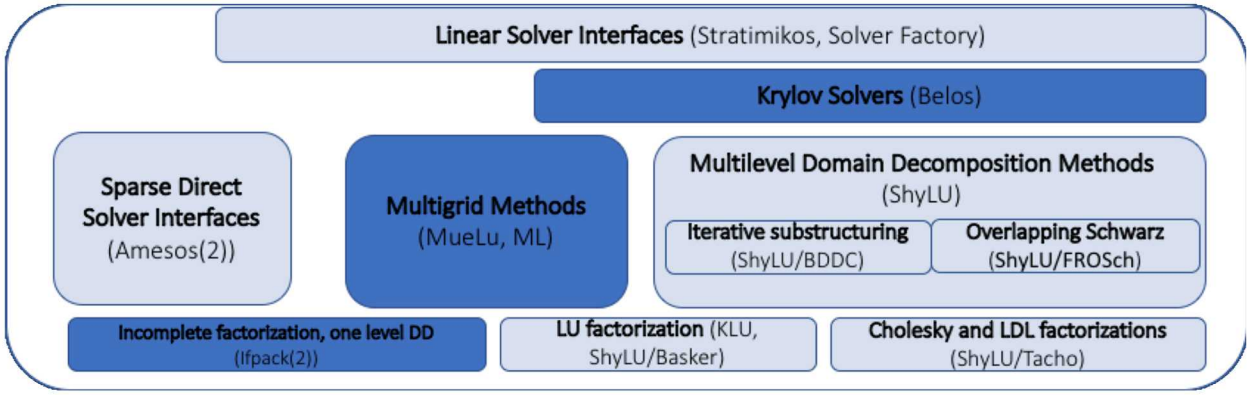


AMG preconditioner is set up only once, and then applied at each time step.

Efficiency of linear preconditioner apply is paramount.

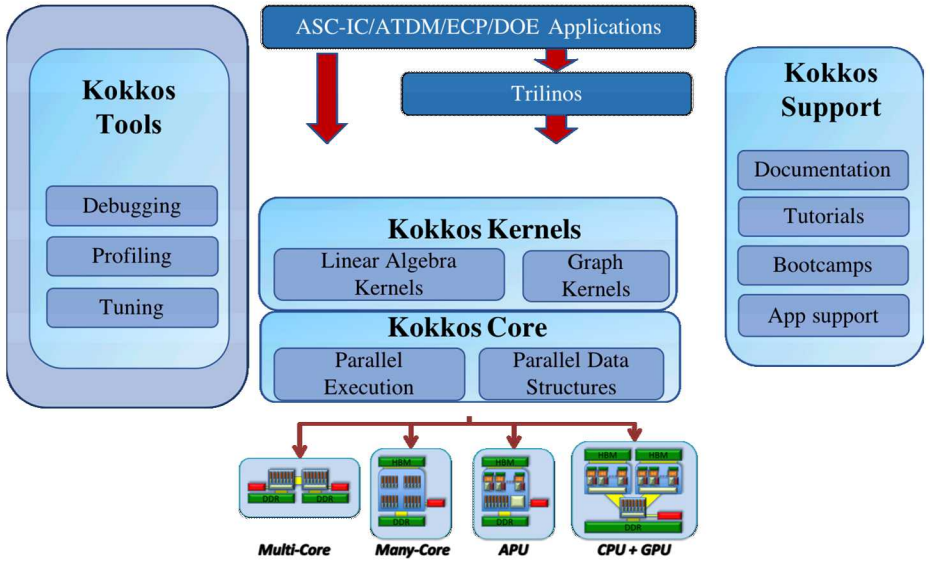
Software

Trilinos Project

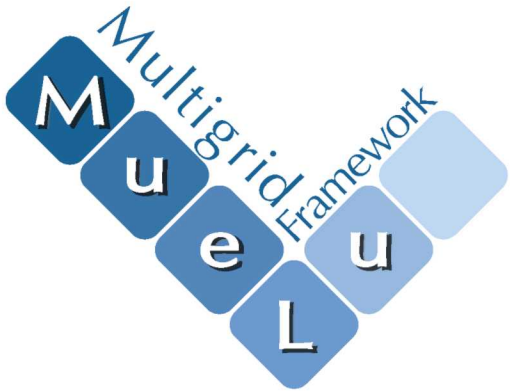


github.com/trilinos/Trilinos

github.com/kokkos



MueLu Multigrid Library



Unstructured algorithms

- classic smoothed aggregation (SA)
- non-symmetric AMG
- AMG for Maxwell's equations

Structured Algorithms

- semi-coarsening AMG
- geometric MG
- structured-grid aggregation-based MG

Leverages many other Trilinos scientific libraries

- Shared memory parallelism from Kokkos → architecture portability
- Sparse distributed linear algebra: Tpetra
- Distributed smoothers: Ifpack2
- Shared memory smoothers, SpGEMM, distance-2 coloring: Kokkos-Kernels
- Load balancing: Zoltan2
- Direct Solvers: Amesos2

Numerical Results

EMPIRE Particle-in-Cell Numerical Experiments

SNL Vortex cluster

- 54 compute nodes
  - Dual socket IBM Power 9, four Nvidia Tesla V100 GPUs
- Interconnect: fat tree Infiniband
- gcc 7.3.1, Spectrum MPI 10.3.0, CUDA 10.1.243
- Application readiness testbed for Sierra supercomputer @ LLNL

Source: SAND-2019-10835R

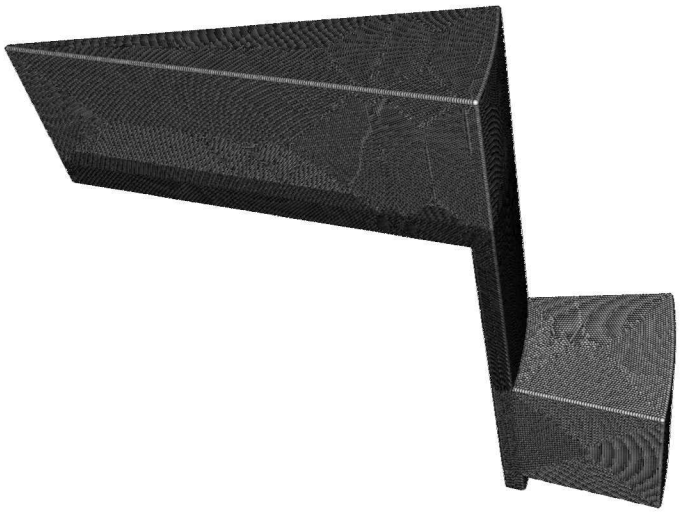
"BDot" mesh

- Small: 10.9m elements, 1.87m nodes
  - Dynamic CFL range: 0.86 to 16.9
- Refined: 85m elements, 14.8m nodes
  - Dynamic CFL range: 0.64 to 16.9

"Cavity" mesh

- 39.1m elements, 6.8m nodes
- Dynamic CFL range: 0.25 to 13.5

1 MPI rank/GPU



Linear Solver Details

Recall the system that we are solving:

$$\begin{pmatrix} \frac{1}{\Delta t}\boldsymbol{M}_B(1) & \boldsymbol{M}_B(1)\boldsymbol{C} \\ -\boldsymbol{C}^T\boldsymbol{M}_B(\mu^{-1}) & \frac{1}{\Delta t}\boldsymbol{M}_E(\epsilon) \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta t}\boldsymbol{M}_B(1) & \boldsymbol{0} \\ -\boldsymbol{C}^T\boldsymbol{M}_B(\mu^{-1}) & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \Delta t\boldsymbol{C} \\ \boldsymbol{0} & \boldsymbol{S}_E \end{pmatrix}$$

$\boldsymbol{S}_E$ solved with Conjugate Gradient + Maxwell AMG

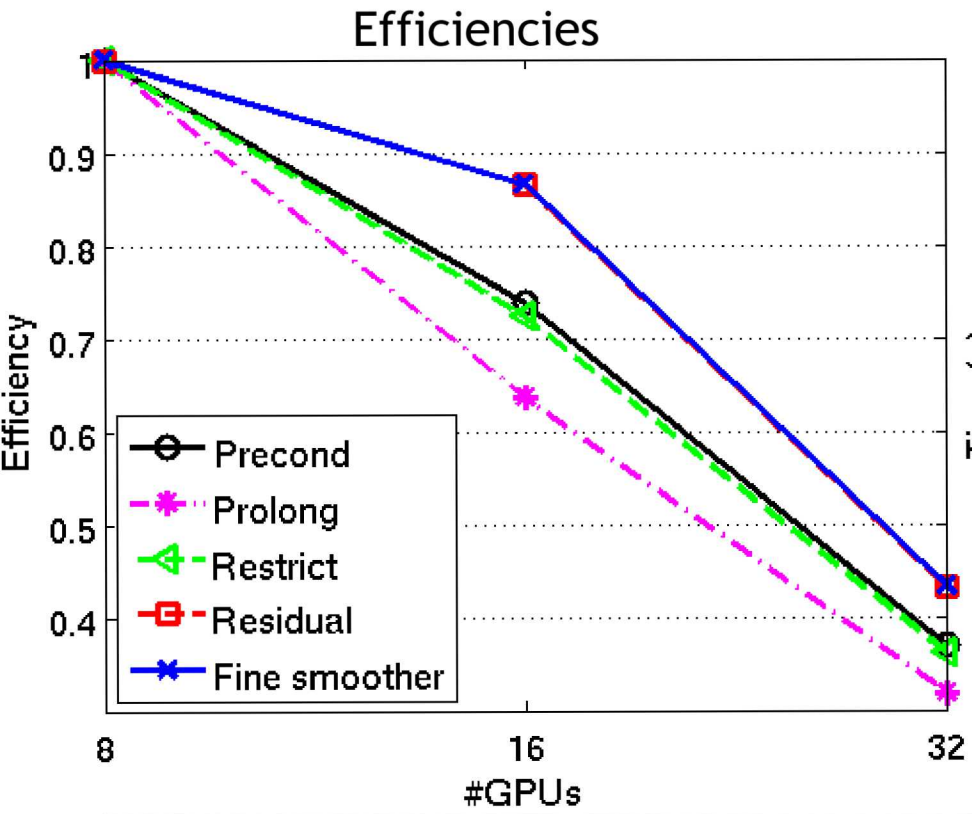
Maxwell AMG preconditioner is set up only once, applied over many timesteps

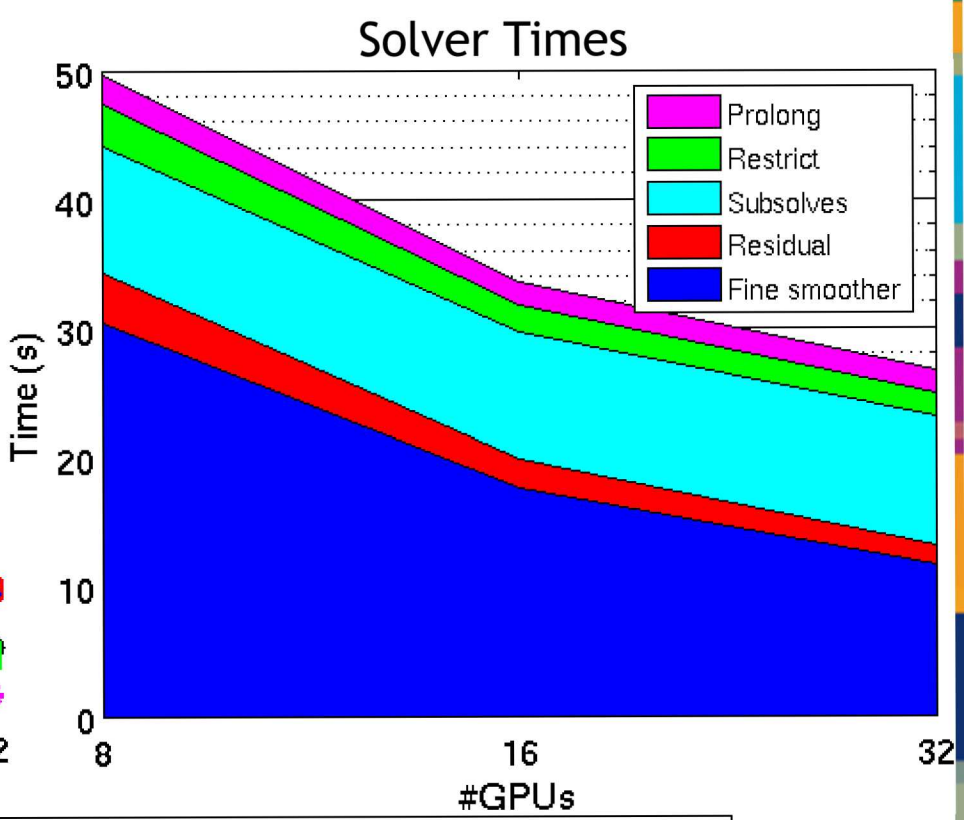
Fine grid smoother: degree 4 Chebyshev polynomial

We will use 2x2 block diagonal (additive) variant

- (1,1) block: single level, degree 6 Chebyshev
- (2,2) block: single level, degree 6 Chebyshev

Bdot: AMG Strong Scaling on "Small" Mesh

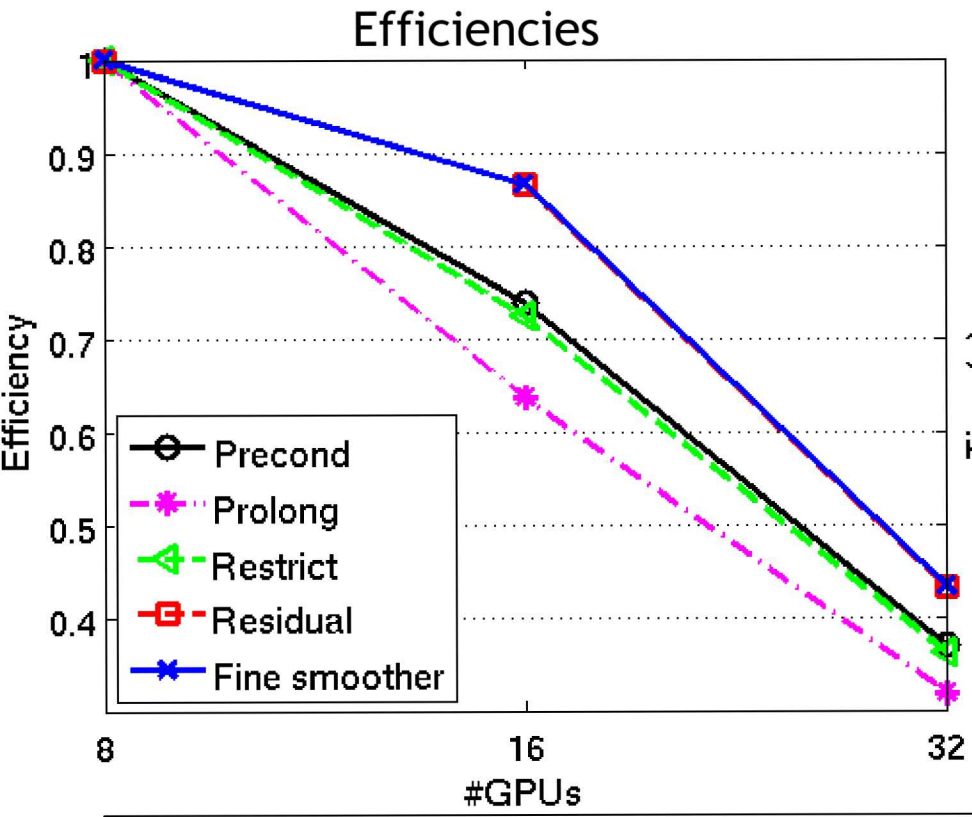


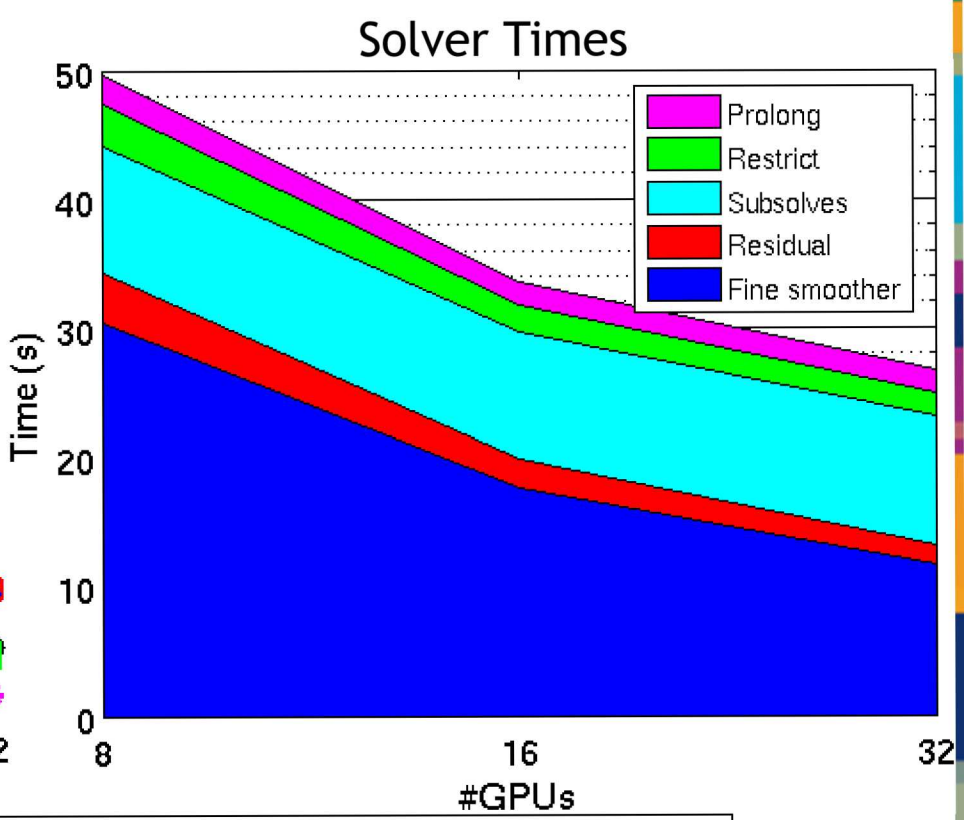


200 timesteps

#GPUs	8	16	32
#Edges/device	1.62M	809k	405k
#Nodes/device	235k	117k	59k

Bdot: AMG Strong Scaling on "Small" Mesh





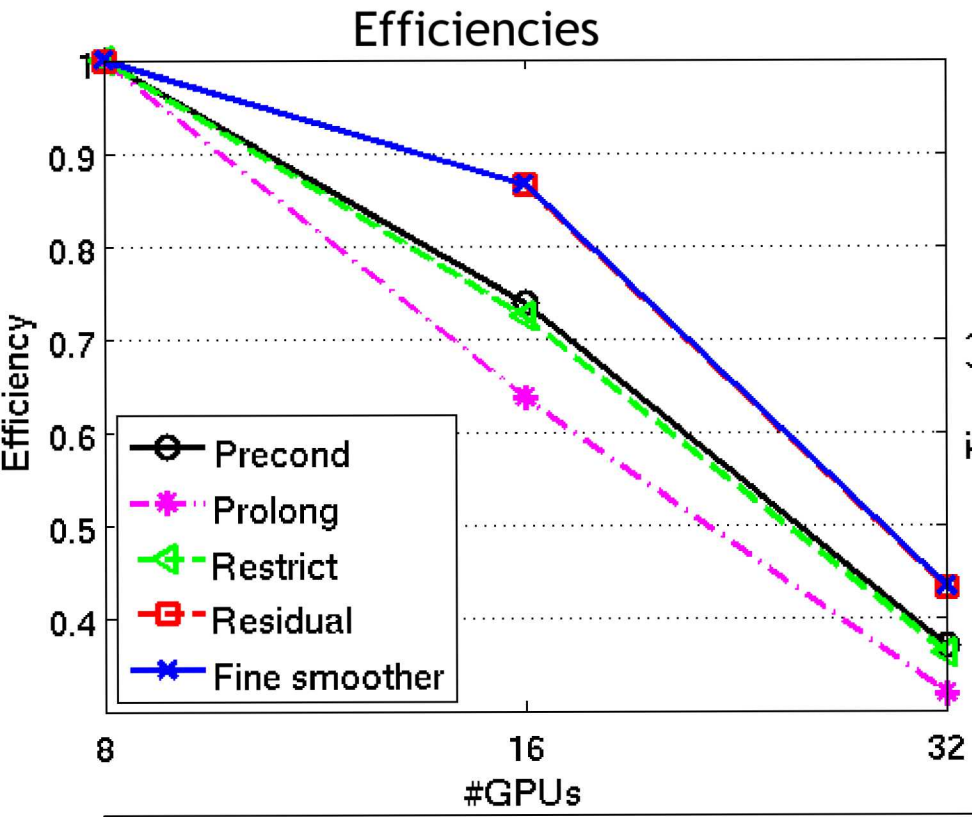
200 timesteps

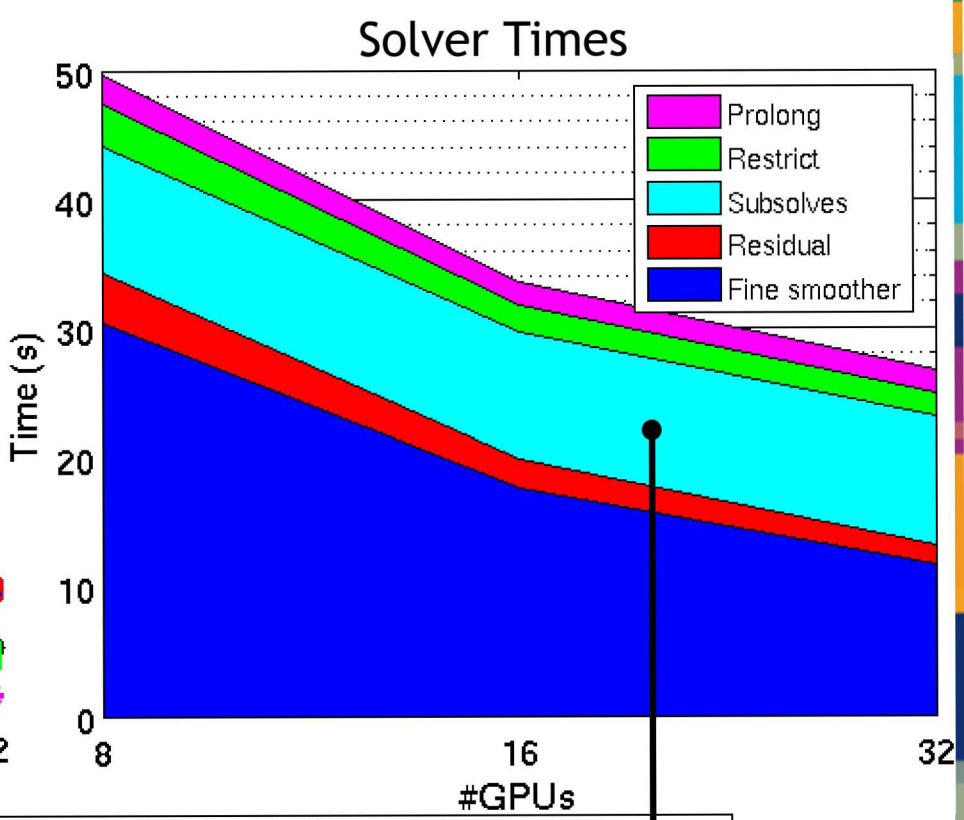
AMG is 70-80% of total linear solve time

6 solves/sec @ 32 GPUs

Chebyshev smoother (matvec kernel) efficiency drops off at @ 32 GPUs

Bdot: AMG Strong Scaling on "Small" Mesh





200 timesteps

AMG is 70-80% of total linear solve time

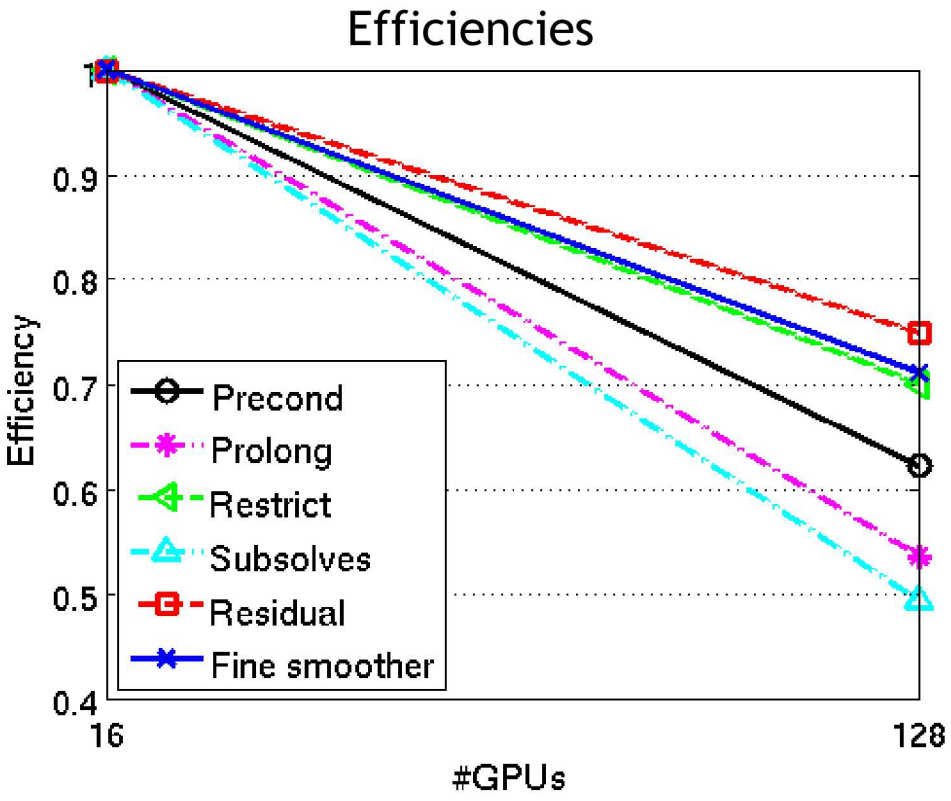
6 solves/sec @ 32 GPUs

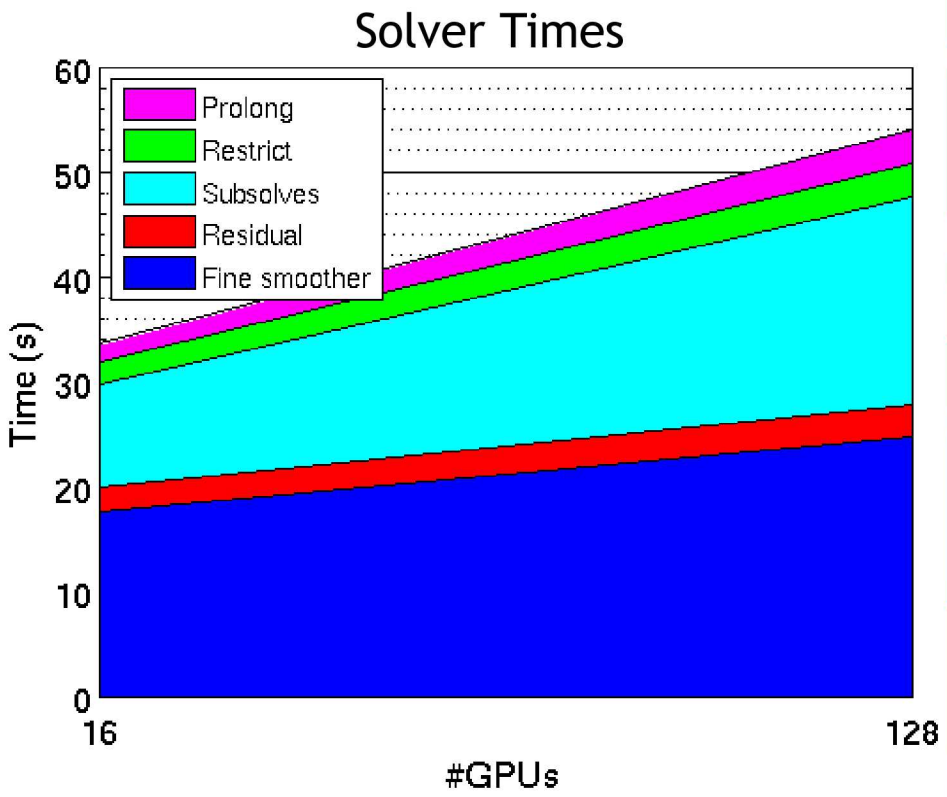
Chebyshev smoother (matvec kernel) efficiency drops off at @ 32 GPUs

Subsolves migrated to small number of devices

- Effect of rebalancing, so no speedup expected

Bdot: AMG Weak Scaling





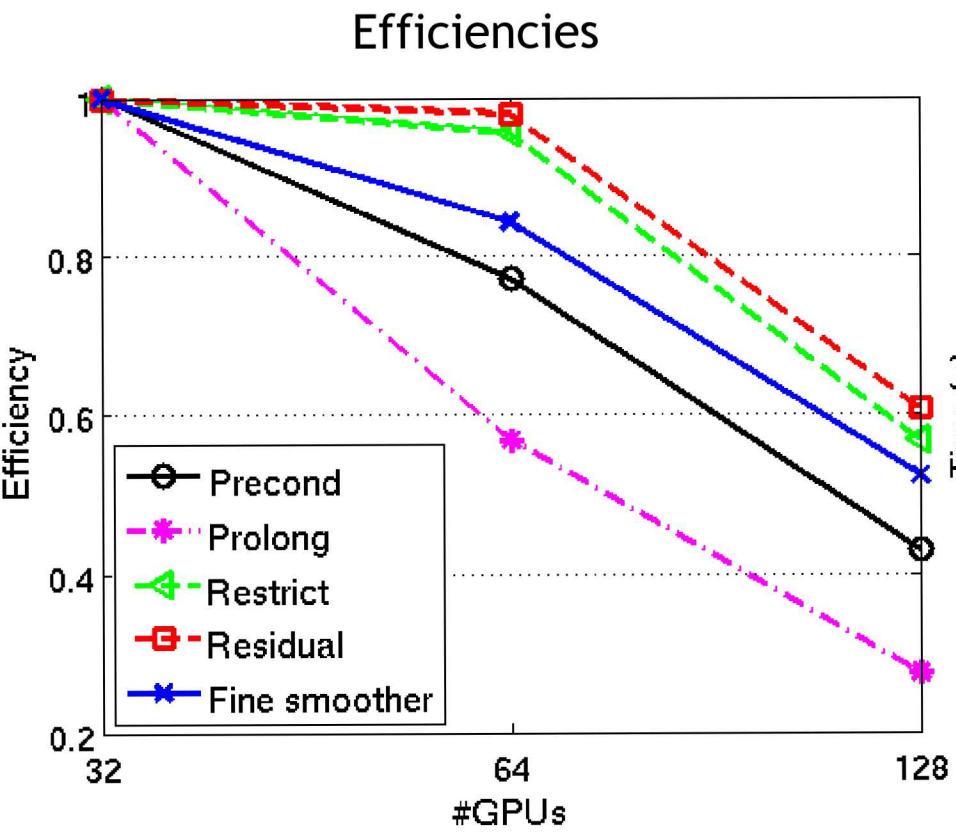
200 timesteps

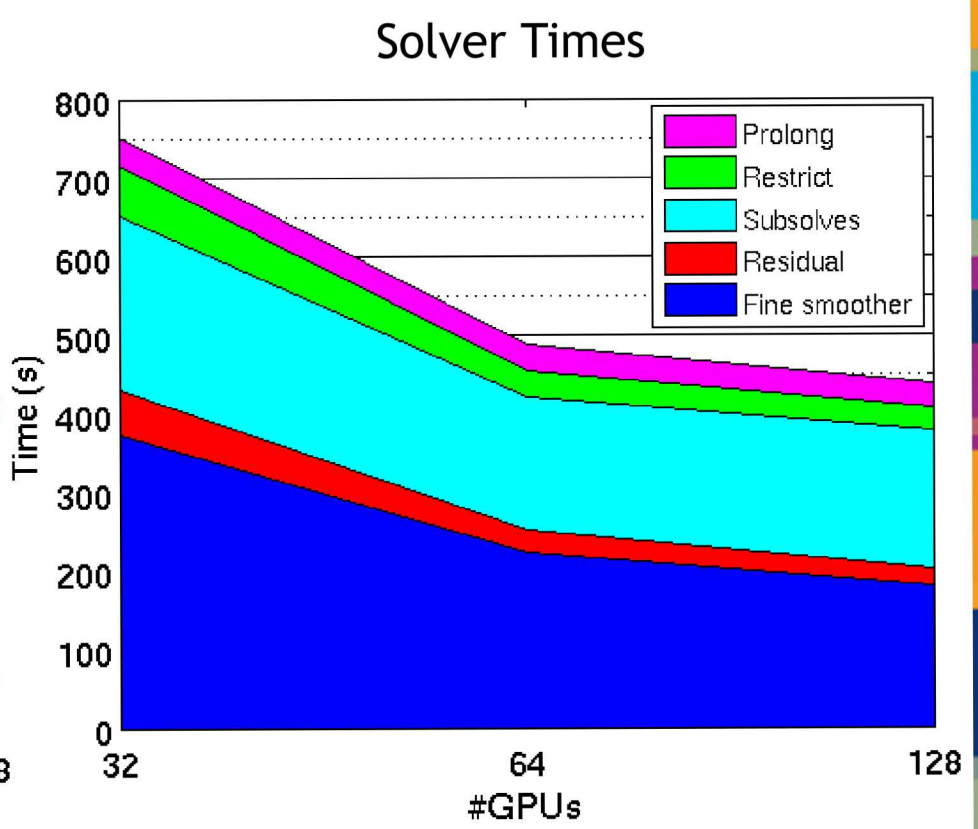
AMG is 75-80% of total linear solve time

4.5 solves/sec @ 16 GPUs

3 solves/sec @ 128 GPUs

Cavity: AMG Strong Scaling

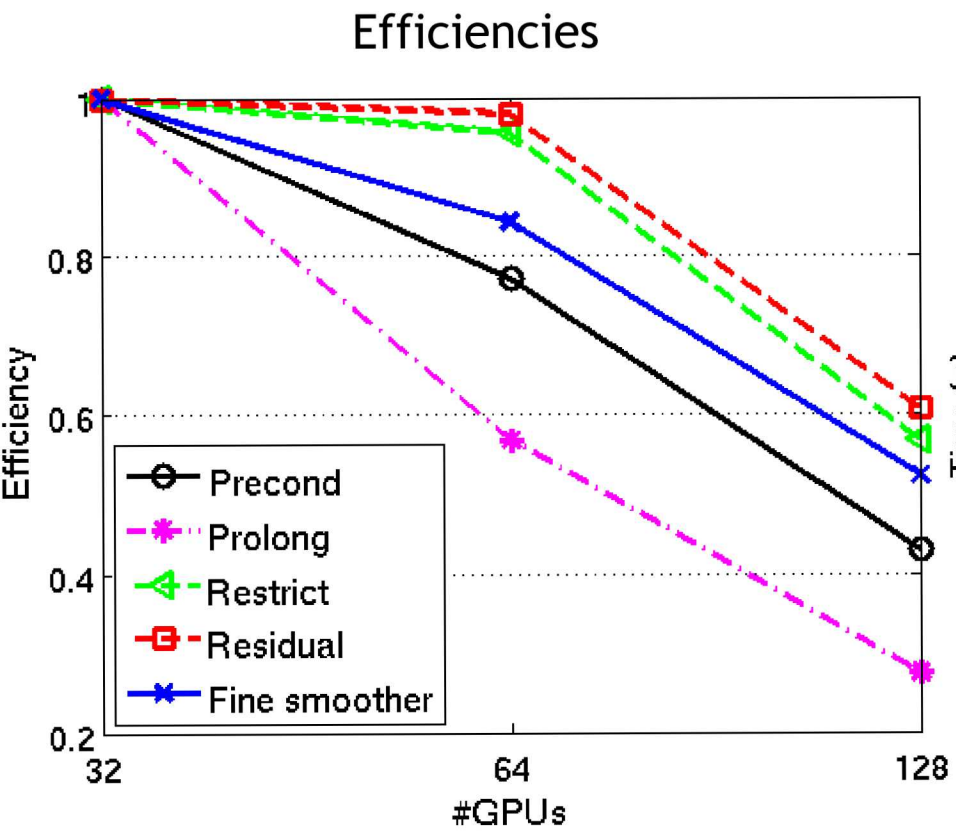


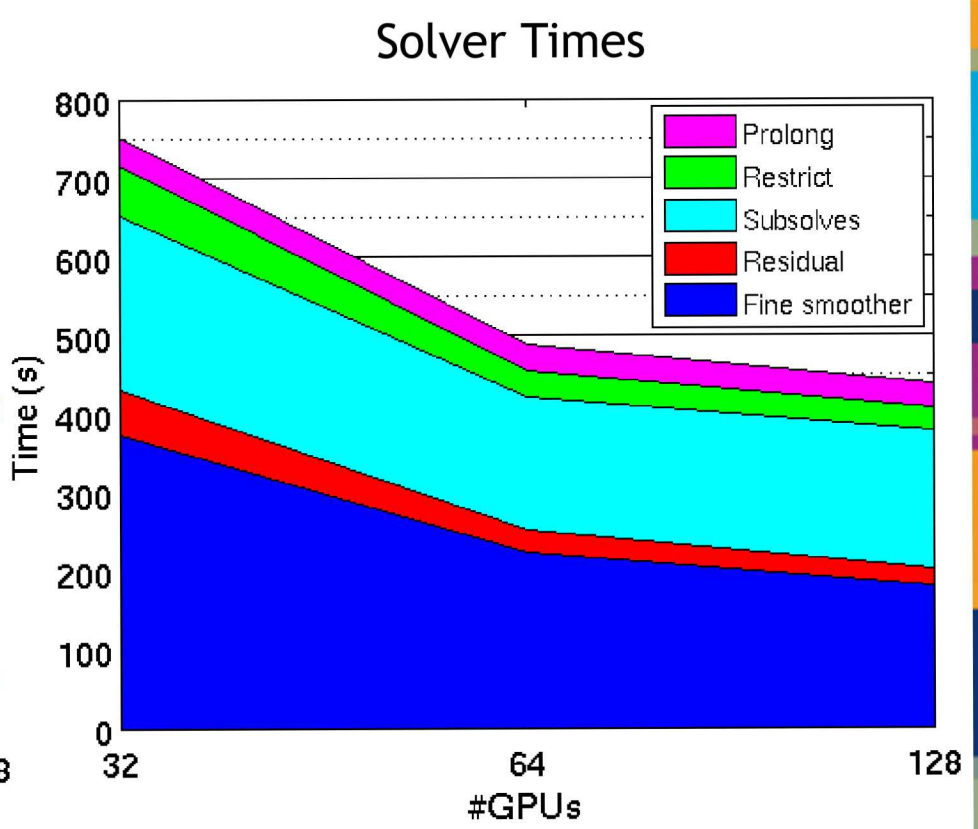


2000 timesteps

#GPUs	32	64	128
#Edges/device	1.46M	728k	364k
#Nodes/device	213k	106k	53k

Cavity: AMG Strong Scaling





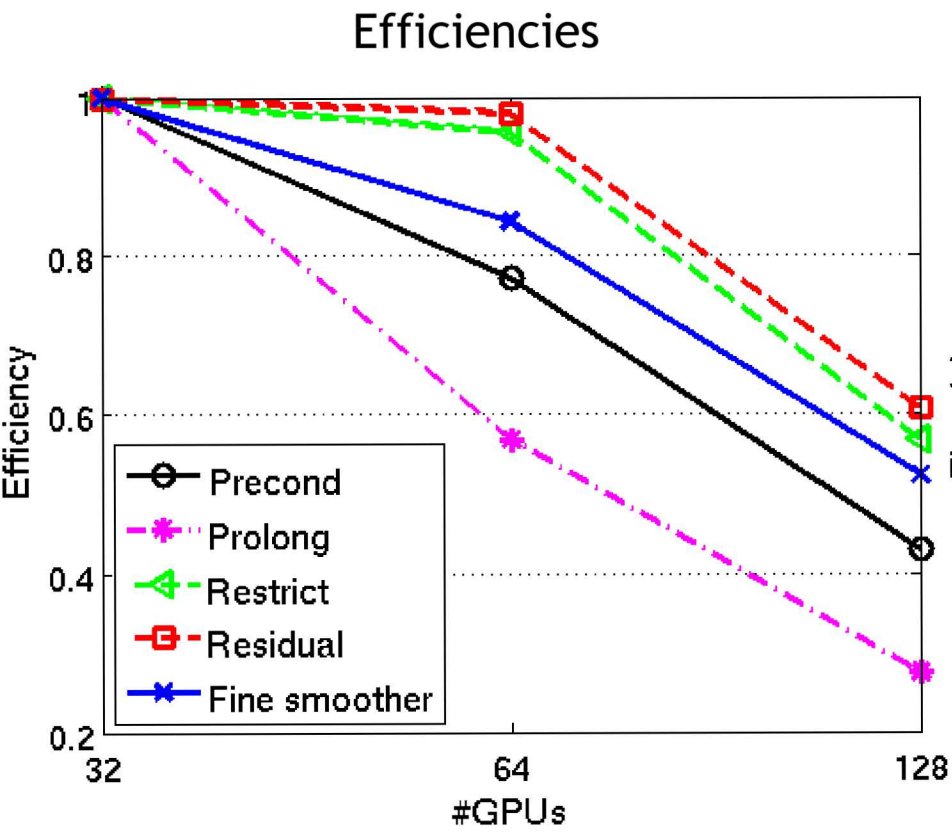
2000 timesteps

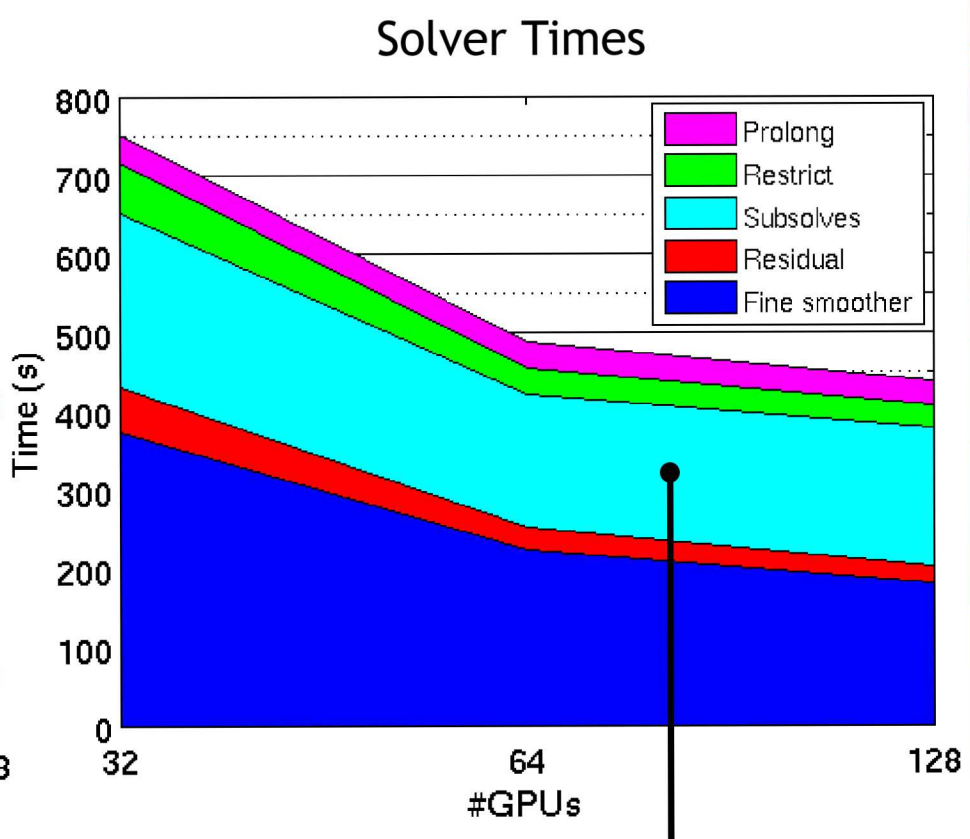
AMG is ~80% of total linear solve time

3.8 solves/sec @ 128 GPUs

At 128 GPUs, subsolve and fine grid smoother are roughly same cost

Cavity: AMG Strong Scaling





2000 timesteps

AMG is ~80% of total linear solve time

3.8 solves/sec @ 128 GPUs

At 128 GPUs, subsolve and fine grid smoother are roughly same cost

Subsolves always occur on same number of devices

- Effect of rebalancing, so efficiency drop expected

Future Work / Concluding Remarks

Assess solver performance on larger cavity meshes on Sierra platform

s-step Chebyshev smoother

- Single halo exchange but with larger message
- Followed by *s* local matrix-vector products

Related talks

- M. Bettencourt, "Towards Exascale Plasma Simulations using PIC Algorithms", Wedn. pm, MS5
- L. Berger-Vergiat, "MueLu's Algorithmic Performance on GPUs", Thursday p.m. MS34
- C. Siefert, "State of the Tpetra Linear Solver Stack", Friday a.m., MS41