Sandia National Laboratories

Exceptional service in the national interest

Linear solvers Update – TUG 2022

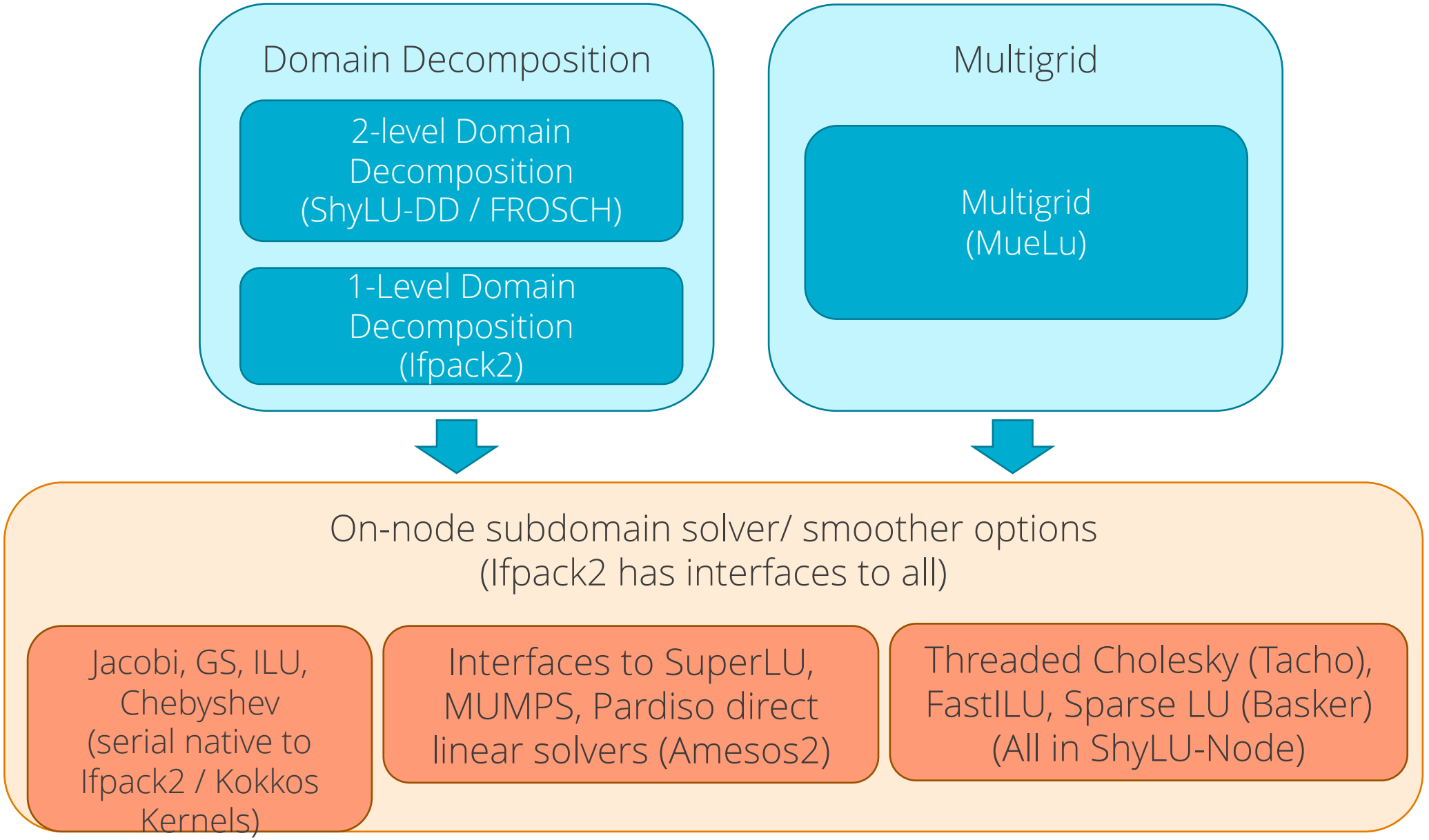
Siva Rajamanickam (Presenter)

Luc Berger-Vergiat, Erik Boman, Vinh Dang, Christian Glusa, Graham Harper, Jonathan Hu, Brian Kelley, Kim Liegeois, Jennifer Loe, Chris Siefert, Ray Tuminaro, Ichi Yamazaki

Trilinos User Group Meeting 2022

SAND2022-14878 PE

Overview of Preconditioning in Trilinos:



Outline

- One Level Domain Decomposition
  - Local preconditioners and Triangular Solve for GPUs
- Batched Linear Solvers
- Multiprecision Solvers
- New Low Synch Orthogonalization
- Basker: Parallel direct solvers for Xyce
- FROSch: Domain Decomposition preconditioners on GPU
- MueLu: New developments in Multigrid methods

Incomplete LU factorization (ILU) Preconditioning Choices for Thermal Fluids Application:

Least Robust

Most Robust

Robustness

Factorization (ILU)

Triangular Solve (TRSV)

Fast ILU	RILU(k)	Serial RILU(k)
Approximate iterative factorization	Extracts some parallelism with level scheduling	Non-parallel
Fast TRSV	**KKSpTRSV**	**Serial TRSV**
Polynomial approximation to triangular solve	Extracts some parallelism with level scheduling	Non-parallel

Parallelism

Faster on GPU

Slower on GPU

ASC Solvers on GPU
Team: Loe, Berger-Vergiat, Dang, Hu, Kelley, Rajamanickam

*FastILU + FastTRSV is faster on GPU *only if* it converges in few enough sweeps.
** KKSpTRSV and Serial TRSV are equally robust, but they may have different numerical behavior.
This is also true for RILU(k) and Serial RILU(k).

RILU(3) + KKSpTRSV (Standard ILU Option)

Historical Speedup on 4 nodes x 4 V100 GPUs:

<u>12.5x Speedup</u> with Metis reordering!

	Trilinos 13.2+ (RCM)	Trilinos Dev (Metis)	Speedup
Compute	10.36	0.98	10.6
Solve	15.71	1.11	14.2
Total	26.07	2.09	12.5

Left: RILU(3)+KKSpTRSV timings from milestone start.
Right: Timings at milestone end. Both on ATS2.

ATS2 (4 V100 GPUs/node) vs
CTS1 (Intel Broadwell 36 cores/node):
<u>1.4x Speedup</u> over fastest CTS1 run.

	ATS2 Best	CTS1 Best	Speedup
Compute	0.98	0.17	0.2
Solve	1.11	2.75	2.5
Total	2.09	2.92	1.4

Left: RILU(3)+KKSpTRSV on ATS2 with Trilinos Dev and Metis reordering.
Right: RILU(3)+KKSpTRSV on CTS1 with Trilinos Dev and RCM reordering.

Improvement Highlights:

- Move some operations from compute (called every solve) to initialize (called with new matrix pattern). (V. Dang)
- Removed extra device to host copies. (V. Dang, J. Hu)
- Performance improvements to Kokkos Kernels RILU(k) numeric (V. Dang)
- Performance improvements to Ifpack2 RILU(k), avoiding extra copies. (B. Kelley, V. Dang, J. Hu)
- Update Ifpack2 interface to allow Metis reordering (I. Yamazaki)

FastILU + FastSpTRSV

**Note: Even though it is speedy, "Fast" ILU and TRSV perform up to 10x more FLOPS than their traditional counterparts! (Cost depends on number of sweeps.)

Historical Speedup on 4 nodes x 4 V100 GPUs:

Infinite speedup! ☺

(FastILU existed in Trilinos but did not build on GPU successfully in Trilinos 13.2.)

16x Speedup over previous fastest ILU!

	Trilinos 13.2 (RCM)	Trilinos Dev Fast ILU	Speedup
Compute	10.36	1.37	7.5
Solve	15.71	0.25	62.5
Total	26.07	1.63	16.0

Left: RILU(3)+KKSpTRSV timings from milestone start on ATS2.
Right: FastILU + FastTRSV on ATS2 with Trilinos Dev and RCM reordering.

ATS2 (4 V100 GPUs/node) vs CTS1 (Intel Broadwell 36 cores/node):

1.8x Speedup over fastest CTS1 run!

	ATS2 FastILU (RCM)	CTS1 Best	Speedup
Compute	1.37	0.17	0.1
Solve	0.25	2.75	10.9
Total	1.63	2.92	1.8

Left: FastILU + FastTRSV on ATS2 with Trilinos Dev and RCM reordering.
Right: RILU(3)+KKSpTRSV on CTS1 with Trilinos Dev and RCM reordering.
(Fastest of all CTS1 ILU-type options.)

Improvement Highlights:

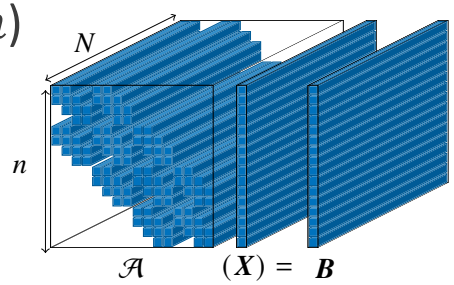
- Fix errors to enable FastILU option for ATS-2 (I. Yamazaki)
- Several performance improvements to FastILU (I. Yamazaki, V. Dang, B. Kelley, S. Rajamanickam)
- Update Ifpack2 interface to allow Metis reordering (I. Yamazaki, E. Boman)
- Coming Soon: “FastILUT” (Based on ParILUT; Chow, Anzt, Dongarra, Rajamanickam, Patel, Boman, others)

Performance Portable Batched Sparse Solvers

Batched Solvers xSDK
PI:S. Rajamanickam
Team: Liegeois, Berger-Vergiat

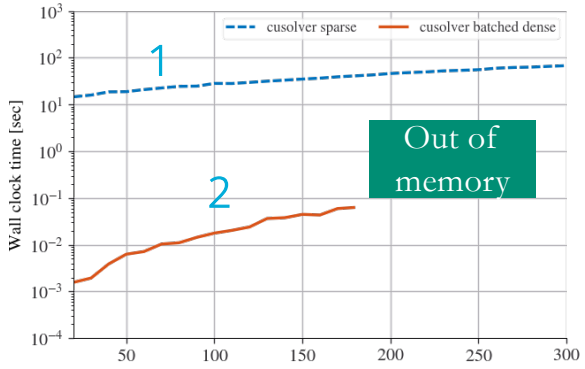
Motivation

Numerical strategies for solving PDEs can lead to large number (N) of small (n) similar **sparse** linear systems to be solved *independently* ($N >> n$)



Two current vendor strategies:

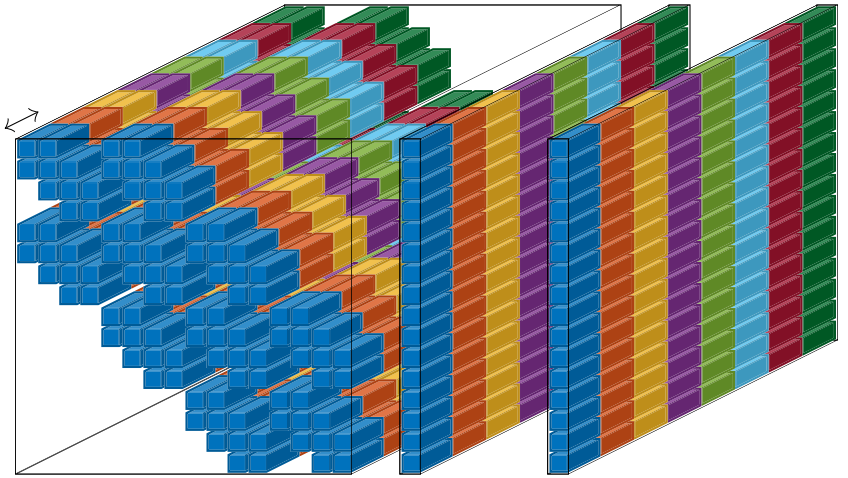
1. Loop over N systems, solve each with vendor sparse solver (**slow**)
2. Convert all systems to dense and use batched dense solver if available (**high-memory footprint**)



New Strategy for Batched Sparse Krylov

- N systems gathered into groups of m systems $m << N$
  - E.g., on new Intel CPU architectures, one can use m=8 and solvers can use vectorization to solve the group of systems at the team level.
- Hierarchical parallelism to solve m systems using an sparse iterative solver at team level

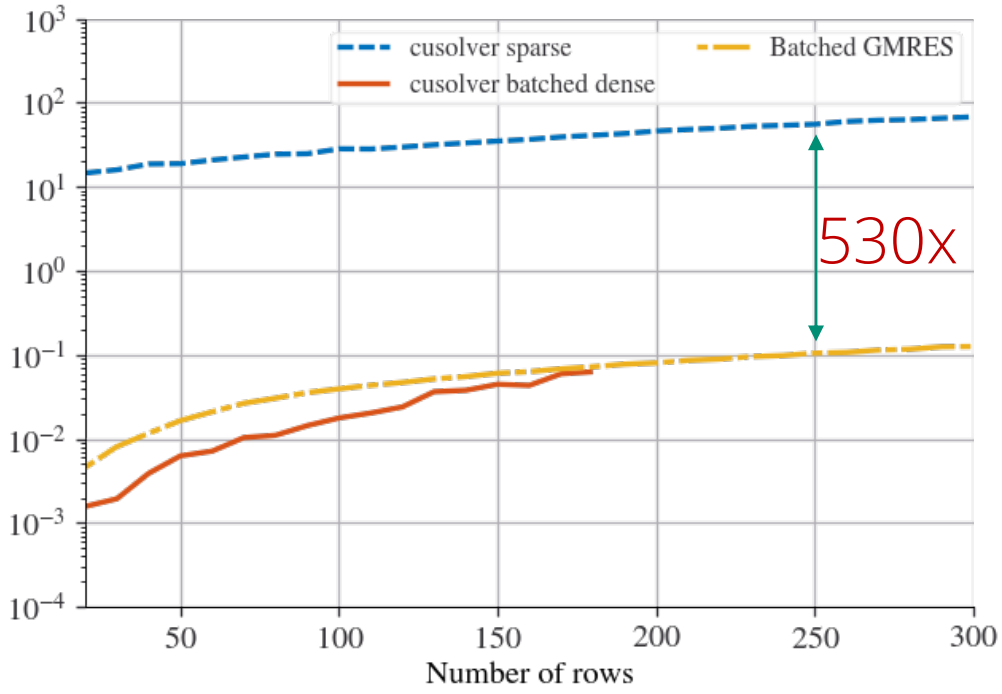
m



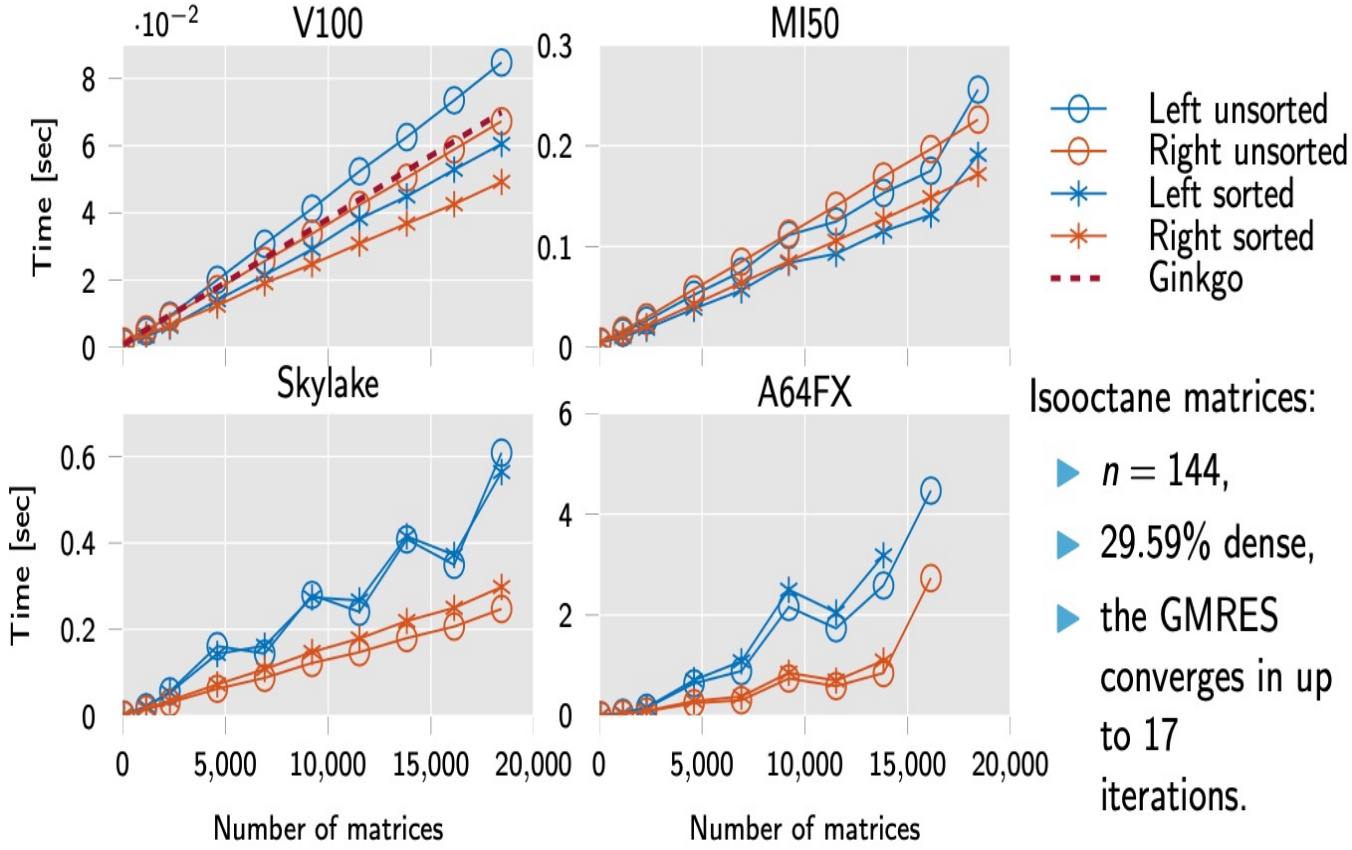
One group/team per color.

Batched sparse solvers implemented using performance portable batched kernels at team level

Batched Sparse Solver (GMRES) Performance Results



Batched GMRES can solve larger problems than batched dense solver from NVIDIA due to reduced memory

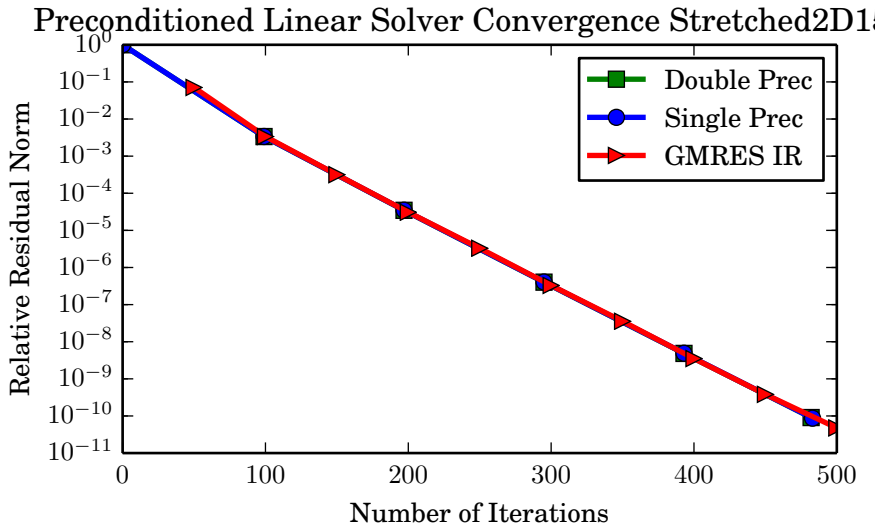


Performance portable to other architecture (same algorithm, different hyperparameters such as m)

Batched sparse solvers efficient use of GPU resources yields two order magnitude speedup versus sequential use of sparse solvers

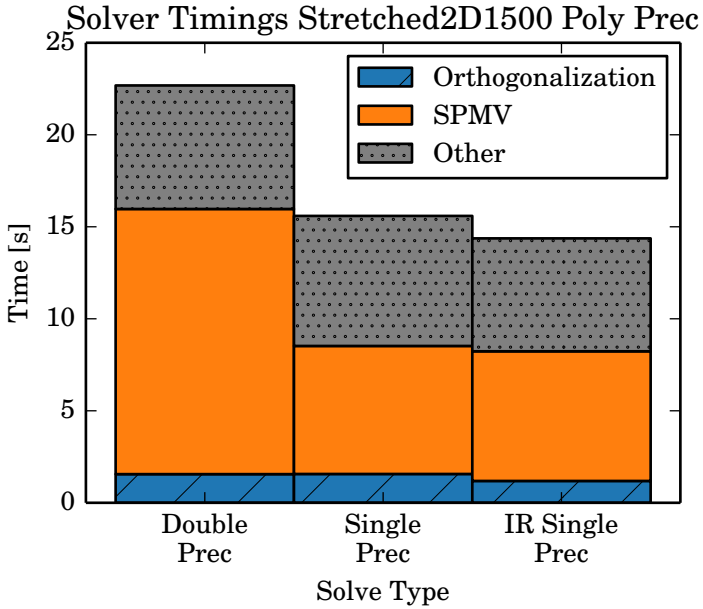
GMRES-IR (Mixed Precision GMRES)

Multiprecision xSDK
PI: S. Rajamanickam
Team: Boman, Loe, Glusa, Yamazaki



- GMRES-IR = GMRES with iterative refinement. Run GMRES + preconditioning in FP32, refine in FP64 to get double-precision accuracy.
- Convergence typically follows double precision GMRES!
- Can also run GMRES-IR with single precision preconditioning. This example: Polynomial preconditioning of degree 40 for a Laplacian.

- About 30% speedup over all-double precision.
- Trilinos options to use single precision preconditioning with double precision solver.
- Stratimikos GMRES-IR interface coming soon!



Coming soon for linear and eigen solvers:

- Stratimikos option to use Belos as a preconditioner for Belos solvers.
  - Allows polynomial preconditioning via Stratimikos.
  - Allows mixed-precision solves such as GMRES-IR (GMRES with iterative refinement).
  - Allows fun combinations like GMRES as a preconditioner for FGMRES.
- Remove Teuchos::SerialDenseMatrix from Belos.
- Belos performance testing and improvements.
- New randomized eigensolver coming to Anasazi.

PEEKS: Low-Synchronous Orthogonalization

Sake: Scalable Kernels for Exascale
PI: Siva Rajamanickam
PEEKS team: Yamazaki, Boman, Bielich

Scope & Objectives

- Low-Synch Orthogonalization: In parallel, Krylov solvers like GMRES are limited by the communication in the orthogonalization step
- Goal: Develop fast (and stable) orthogonalization methods. Integrate in GMRES in Trilinos/Belos
  - Low-synch (delayed) classical Gram-Schmidt with reorth. (DCGS2)
  - Integrate into Trilinos for wide distribution to ECP users

Results on Summit (180 GPUs)

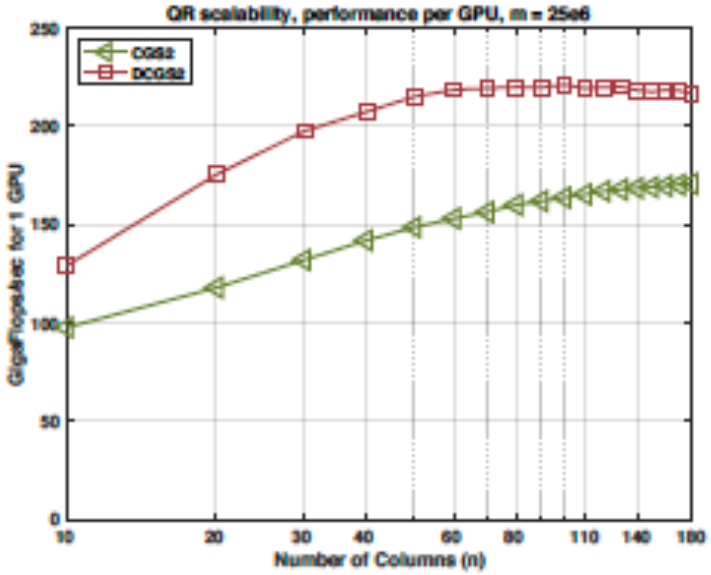


Figure from Bielich et al. (2021)

Accomplishments

Algorithms: Enhanced single-reduce orthogonalization methods in collaboration with Stephen Thomas (ExaWind,NREL) and Bielich/Langou (CU Denver). Extended to block version.

Parallel/GPU: The code has been tested and optimized in multi-GPU environments (Summit), on up to 180 GPUs.

Speedup: The new DCGS2 is 40-50% faster than the old CGS2.

Software: Implemented new methods as option in the Trilinos package Belos. Works on both CPU and GPU. Delivered block variant for s-step GMRES in Trilinos.

Paper: Bielich et al., accepted in Parallel Computing.

Potential Impact

- Low-synch orthogonalization makes GMRES faster and more scalable by reducing parallel communication/synchronization.
- May also speed up dense QR factorization.
- All ECP applications that use Trilinos solvers could benefit, particularly on GPU architectures.

Random sketching (work in progress)

Fast orthogonalization based on matrix sketching

- Work based on Balabanov & Grigori (2021)
- A random sketch projects into a smaller subspace, thus reducing both memory and computational cost
- Most useful for dense linear algebra, such as dense QR
- Pursuing a new QR algorithm based on sketch GMRES followed by CholQR

sGMRES (sketched GMRES)

- Based on method by Nakatsukasa & Tropp (2022)
- Often faster but less robust than standard GMRES
- Works best on nearly symmetric problems,

ShyLU-Basker Linear Solver

- It is primarily designed for circuit analysis (Xyce)
  - Thread-parallel version of KLU
- It uses Block Triangular Form (BTF) typical in circuit matrices
  - Reduces the computational & storage costs (only the diagonal blocks are factored)
  - Increases the thread scalability (all the diagonal blocks can be factored in parallel)
    - Assigns a thread to each "small" block
    - Uses nested-dissection to factor a larger block with multiple threads
- Several extensions were made to improve the thread-parallel stability and performance
- Iterative refinement is implemented in Amesos2 to recover from potential accuracy loss

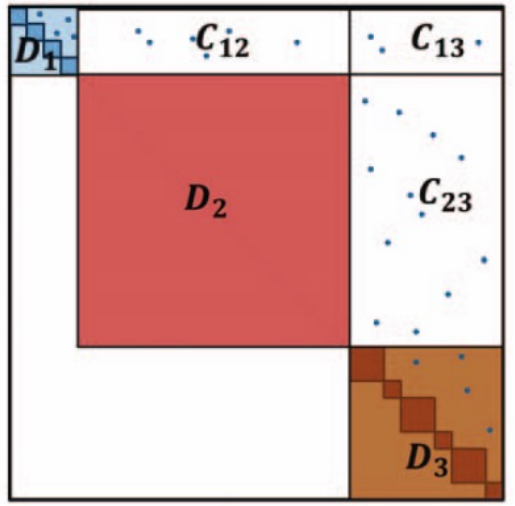
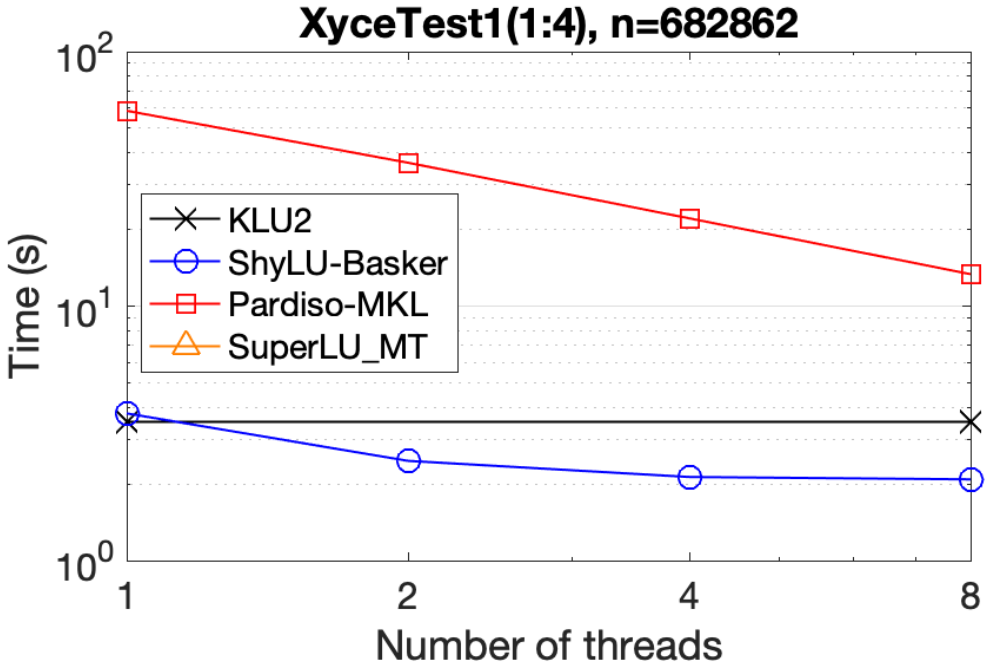


Image taken from "Basker: a threaded sparse LU factorization utilizing hierarchical parallelism on data layouts", J. Booth, S. Rajmanickam, H. Thornquist, IPDPSW, 2016

Current work

- Transpose solve to avoid factoring both A and A^T
- Investigating further improvement in performance & stability
  - Supernodal approach (e.g., Pardiso) may be preferred if no BTF
  - Potential instability of threaded factorization
- Running full Xyce simulation with ShyLU-Basker



ASC: Linear Solvers for Xyce
PI: Ichi Yamazaki
Team: Nathan Ellingwood, Heidi Thornquist

Project: FASTMath, Sake
PI: Ichi Yamazaki , Siva Rajamanickam
Team: Heinlein

Fast and Robust Schwarz (FROSch) Preconditioners

FROSch implements **two-level Schwarz domain decomposition preconditioners** of the form:

$$M_{OS2}^{-1} = \Phi A_0^{-1} \Phi^T + \sum_{i=1}^{N} R_i^T A_i^{-1} R_i$$

The coarse basis is constructed **algebraically from the fully assembled and parallel distributed system matrix** using **operator-based extensions**. FROSch preconditioners have shown to be **robust and scalable for many challenging problems**, including coupled velocity-temperature problems for non-uniform Greenland meshes

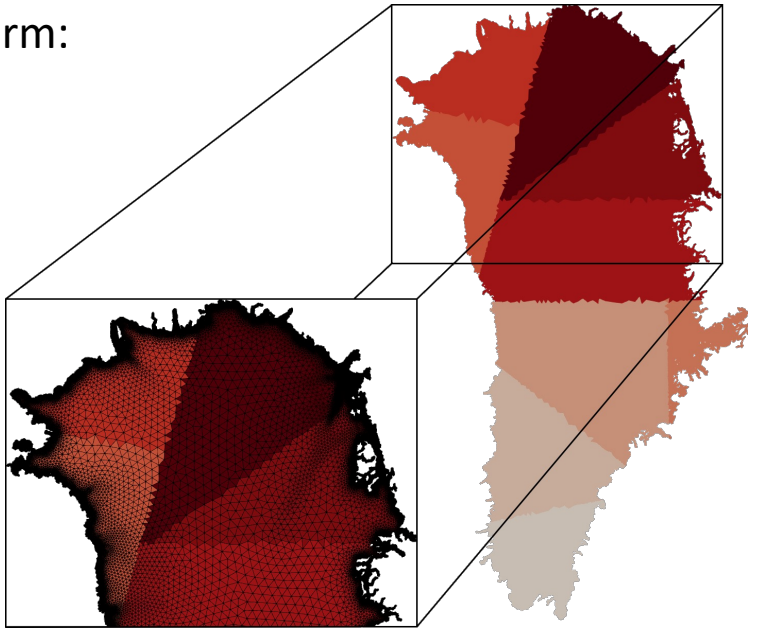


Image taken from "FROSch preconditioners for land ice simulation of Greenland and Antarctica", A. Heinlein, M. Perego, and S. Rajmanickam, SISC, 2022

Highlight features

- Algebraic construction allows for using FROSch itself as an inexact coarse solver
  → **multilevel Schwarz preconditioners**
- Can be used for single-physics systems as well as multi-physics block systems
  → **monolithic Schwarz preconditioners**
- Flexibility with respect to solving the local overlapping and coarse problems:
  Interfaces to Amesos2 (sparse direct), Ifpack2 (incomplete LU), Belos (preconditioned Krylov), Stratimikos
- Interface from Stratimikos, and used as preconditioner for Belos

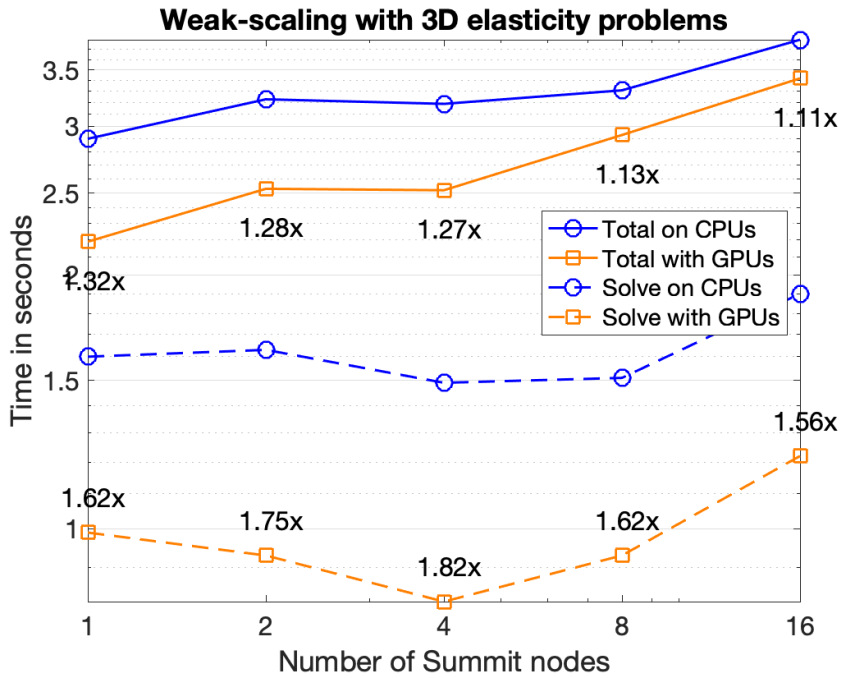
Sandia National Laboratories

GPU Capabilities for FROSch

- Software capability has been extended for **improved GPU performance**
  - **Restructuring parts of FROSch** to further **separate the symbolic and numerical setups**
  - Integratation of Kokkos-Kernels *supernodal* sparse-triangular solve with SuperLU/Cholmod
    [I. Yamazaki, S. Rajmanikam, N. Elingwood, ICPP'20]
  - Extended Trilinos sparse direct solver interface (Amesos2 to Tacho) to use perform the **local solves on GPUs**
  - New options for **approximate/inexact local and coarse solvers**
    (Fast ILU/SpTRSV, FROSch as HalfPrecision operator)

Results

- Similar setup time on CPUs and GPUs, but **2x faster solve time on GPUs** for solving 3D elasticity problems on Summit
  (using 42 CPU cores vs. 6 GPUs / node)
- Next step: testing the performance with Albany and on land ice simulations for Antarctica

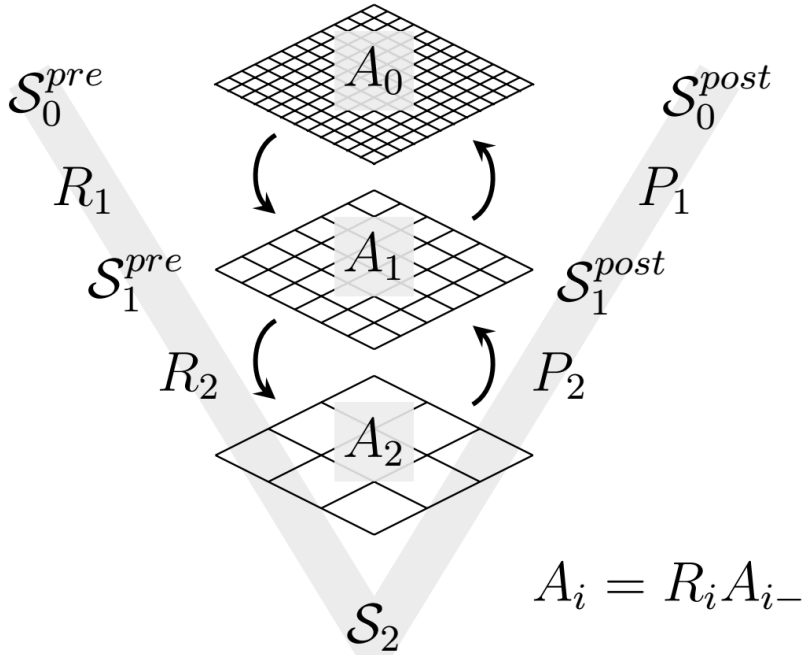


Sandia National Laboratories

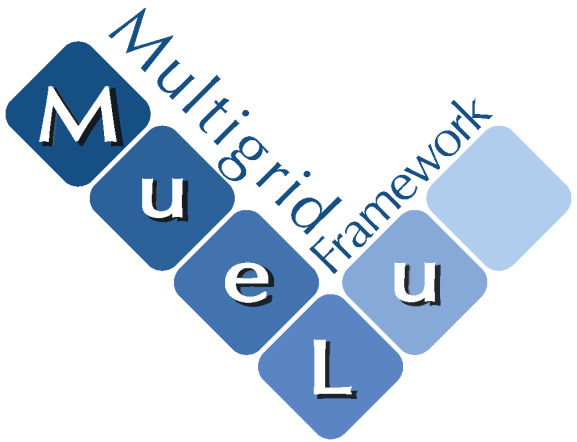
MueLu

Project: Multiple projects
PI: Jonathan Hu
MueLu team: Berger-Vergiat, Glusa, Harper, Seifert, Tuminaro

- MueLu is a multigrid solving/preconditioning package.
- Part of the second-generation of Trilinos
  - Templated on scalars, ordinals, and nodes
- Multigrid is an optimal complexity O(n) solver for linear systems.
  1. Start with a "fine grid"
  2. Smooth error, transfer to coarser grid
  3. Repeat 2...
  4. Perform a direct solve on the "coarse grid"
  5. Transfer to finer grid, smooth error
  6. Repeat 5...
  7. Transfer to original "fine grid", smooth error
- "Is multigrid right for me?"
  - When backslash doesn't cut it



$$A_i = R_i A_{i-1} P_i$$



MueLu Capabilities

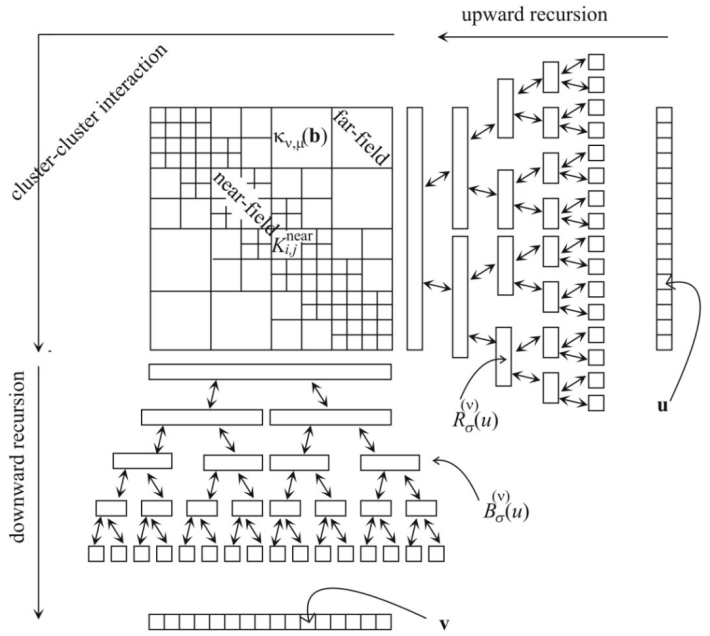
- Can precondition a linear system or iteratively solve a linear system
- Supports a wide variety of grid transfers, smoothers (via Ifpack2/Amesos2), and more
- Inputs commonly supplied in XML format
  - Updated MueLu tutorial: https://muelu-tutorial.readthedocs.io (subject to change)

```
<ParameterList name="MueLu">

  <Parameter name="verbosity" type="string" value="high"/>

  <Parameter name="max levels" type="int" value="3"/>
  <Parameter name="coarse: max size" type="int" value="10"/>

  <Parameter name="multigrid algorithm" type="string" value="sa"/>

  <!-- Smoothing -->
  <!-- Comment/uncomment different sections to try different smoothers -->

  <!-- Jacobi -->
  <Parameter name="smoother: type" type="string" value="RELAXATION"/>
  <ParameterList name="smoother: params">
    <Parameter name="relaxation: type"  type="string" value="Jacobi"/>
    <Parameter name="relaxation: sweeps" type="int" value="1"/>
    <Parameter name="relaxation: damping factor" type="double" value="0.9"/>
  </ParameterList>

  <!-- Aggregation -->
  <Parameter name="aggregation: type" type="string" value="uncoupled"/>
  <Parameter name="aggregation: min agg size" type="int" value="3"/>
  <Parameter name="aggregation: max agg size" type="int" value="9"/>

</ParameterList>
```

- Required packages: Teuchos, Xpetra, KokkosCore*, KokkosContainers*, KokkosKernels*
- Optional packages: Belos, Epetra, Teko, Amesos(2), Ifpack(2), Intrepid2, ML, Tpetra, Zoltan(2), Stratimikos, Thyra

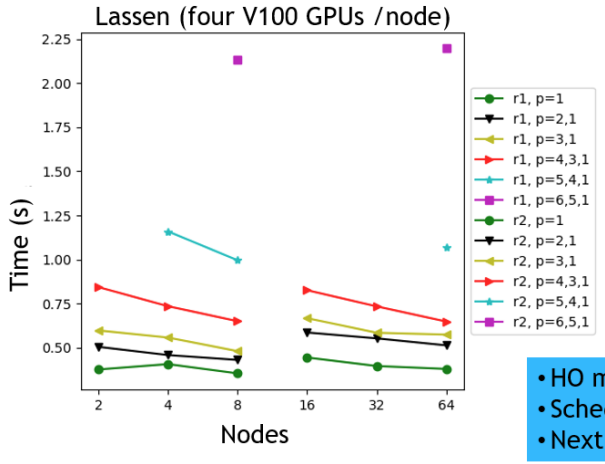
MueLu Developments 1

- Hierarchical Matrices
  - For applications with dense matrices
- Maxwell
  - Multigrid for electromagnetics
- Multiprecision
  - Do restriction and prolongation apply at lower precision than finer levels
- Higher Order
  - P-coarsening
  - Supports schedules



Unit cube domain, uniform tetrahedral meshes

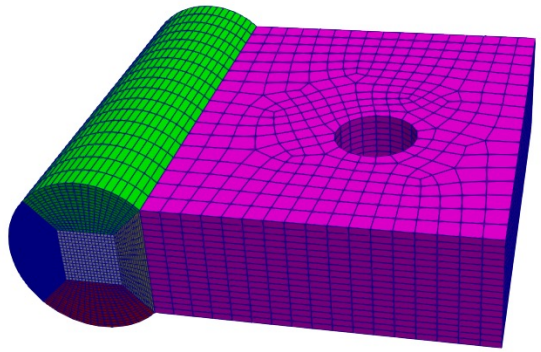
Two problem sizes: R1 @ 2.3M DOFs, R2 @ 18.7M DOFs; one time step.



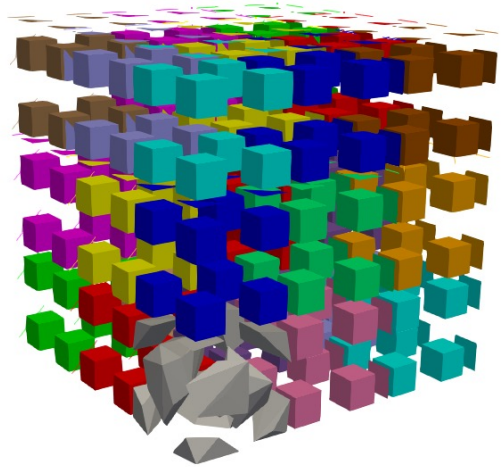
p coarsening schedule	nnz/row	Iterations	
		R1	R2
1	16	30	31
2 -> 1	43	24	24
3 -> 1	84	27	26
4 -> 3 -> 1	144	21	20
5 -> 4 -> 1	224	32	32
6 -> 5 -> 1	328	63	63

- HO multigrid effective in *h* and *p*
- Scheduling experiments limited by memory for p=5,6
- Next steps: integrate into EMPIRE, optimize, matrix-free

MueLu Developments 2



- Region Multigrid
  - Grids of grids
  - Subgrids coarsened geometrically or algebraically
- Geometric Multigrid
  - Problems with structured meshes
- Support for BlockCrs matrices throughout AMG hierarchy
- Aggregation (formation of coarse level DOFs)
  - Pairwise Aggregation
  - “Cut Drop” strength-of-connection
    - For dropping weak distant connections during aggregate formation
    - Addresses matrix fill-in seen in standard Sa-AMG



Ongoing MueLu Developments & More

- Machine Learning for AMG
  - Determining tolerances via ML

	Heuristics			Fixed	
Heuristic	1	2	3	0	0.025
1		-26	11	182	-23
2	26		37	182	0
3	-11	-37		175	-33
0	-182	-182	-175		-170
0.025	23	0	33	23	

- Matrix-Free
  - Matrix-free tentative prolongator operator
  - Hierarchy treats grid transfers as operators
  - Synergistic with high order work and Maxwell

Heuristic	disk1	disk2	disk3	expl	tubes
1	37	21	21	30	14
2	22	21	21	30	14
3	37	26	21	30	14
fixed 0	57	28	24	30	13
fixed 0.01	29	26	31	26	14
fixed 0.025	22	21	21	22	14

- Patch-based smoothers
  - Addresses algorithmic gap between point SOR methods and incomplete factorizations

Summary

- Several new developments in new algorithms.
- Several options available for accelerators.
- Several examples to demonstrate productionizing the algorithm capabilities that were developed.
- Exciting times ahead.