Exceptional service in the national interest

Linear Solvers and Preconditioners

EuroTUG 2022

Graham Harper, Jennifer Loe

Center for Computing Research

Sandia National Laboratories

14/08/22

SAND2022-12312 C

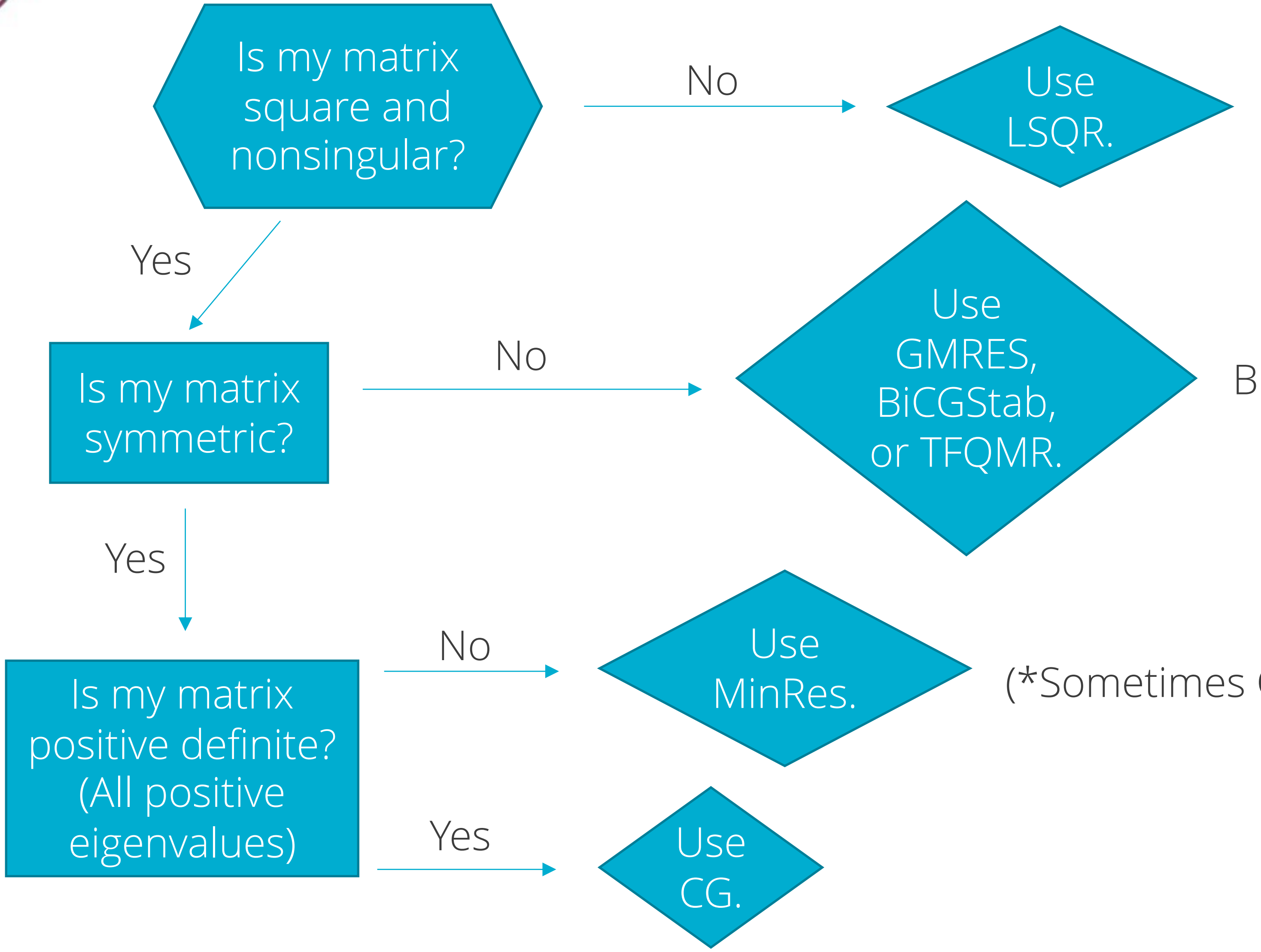
Linear Solver and Preconditioning Packages in Trilinos

- Belos – Iterative solvers
- Ifpack2/ShyLU – Factorization-based and domain decomposition preconditioners
- Amesos2 – Direct solvers interface
- MueLu - Multigrid

Belos: Iterative and Krylov solvers + Preconditioning

1) Belos Basics:
    1) How to choose the best solver for your needs.
    2) Common solver parameters
2) New(ish) Belos Capabilities:
    1) Communication-Avoiding Tpetra solvers
    2) Upcoming Mixed precision GMRES
3) Factorization-Based Preconditioners
    1) Overview of Capabilities
    2) Recent performance improvements

Choosing a Belos solver for Ax=b: (One right-hand side.)


Is my matrix square and nonsingular?
No
Use LSQR.
Yes
Is my matrix symmetric?
No
Use GMRES, BiCGStab, or TFQMR.
Yes
Is my matrix positive definite? (All positive eigenvalues)
No
Use MinRes.
Yes
Use CG.

But how to choose? See next slide.

(*Sometimes CG still works for indefinite matrices.)

Comparing Belos Solvers for Non-Symmetric Linear Systems

Costs for kth iteration:

	SpMV	axpy	Inner Product	Stored vectors
GMRES	1	k+1	k+1	k+5
TFQMR	2	10	4	8
BiCGStab	2	6	4	10

(Table adapted from: A Comparison of Preconditioned Krylov Subspace Methods for Large-Scale Nonsymmetric Linear Systems. Ghai, Lu, Jiao.)

GMRES is tried and true, with theoretical convergence guarantees minimizing $||b - Ax||_2$, but its convergence can be slowed by restarting.

TFQMR and **BiCGStab** can be much cheaper than GMRES when GMRES needs many iterations. However, they may have more erratic convergence.

Alternative for non-symmetric systems that need many iterations and restarts in GMRES: Use **GCRO-DR** with large (25-50) "**Num Recycled Blocks**." This option will retain important eigenvector information from the Krylov subspace at the restart to improve convergence.

What about multiple right-hand sides (RHS)?

Block vs PseudoBlock methods:

- Both can solve for a single RHS or multiple. (Recommend PseudoBlock for a single RHS.)
- **Block methods** use the information from *ALL right-hand sides* mathematically to find the best solution for each right-hand side. This can sometimes give faster convergence and/or better solutions, but it comes at an extra computational cost.
- **PseudoBlock methods** are *mathematically equivalent* to solving for each right-hand side individually, but they provide improved performance by applying certain mathematical operations to all right-hand sides concurrently.
- Additionally, **recycling methods** can be helpful for a problem with a sequence of right-hand sides; they use information from the previous RHS to help with the next RHS.

Block Methods	PseudoBlock Methods	Single RHS Only Methods	Recycling Methods (1 RHS at a time; good for sequences)
BlockCG	PseudoBlockCG	MinRes	RCG (Recycling CG)
BlockGMRES	PseudoBlockGMRES	TFQMR	GCRODR (GMRES with recycling)
BlockGCRODR	PseudoBlockTFQMR	LSQR	

Translating Common Belos Solver Parameters:

- **Block Size:** (Typically) number of right-hand sides in linear system.
- **Maximum Iterations**: Maximum number of solver iterations.
- **Orthogonalization**: ICGS[2] (2 steps of blocked Classical Gram-Schmidt), IMGS[1] (Modified Gram-Schmidt), DGKS (Gram-Schmidt using tolerance to determine need for re-orthogonalization). [Or TSQR (communication-avoiding Tall-Skinny QR) if enabled.]
- **Convergence Tolerance:** Residual norm tolerance for $||b - Ax||_2$.
- **Num Blocks**: Restart length for GMRES.

Can I use a variable preconditioner?

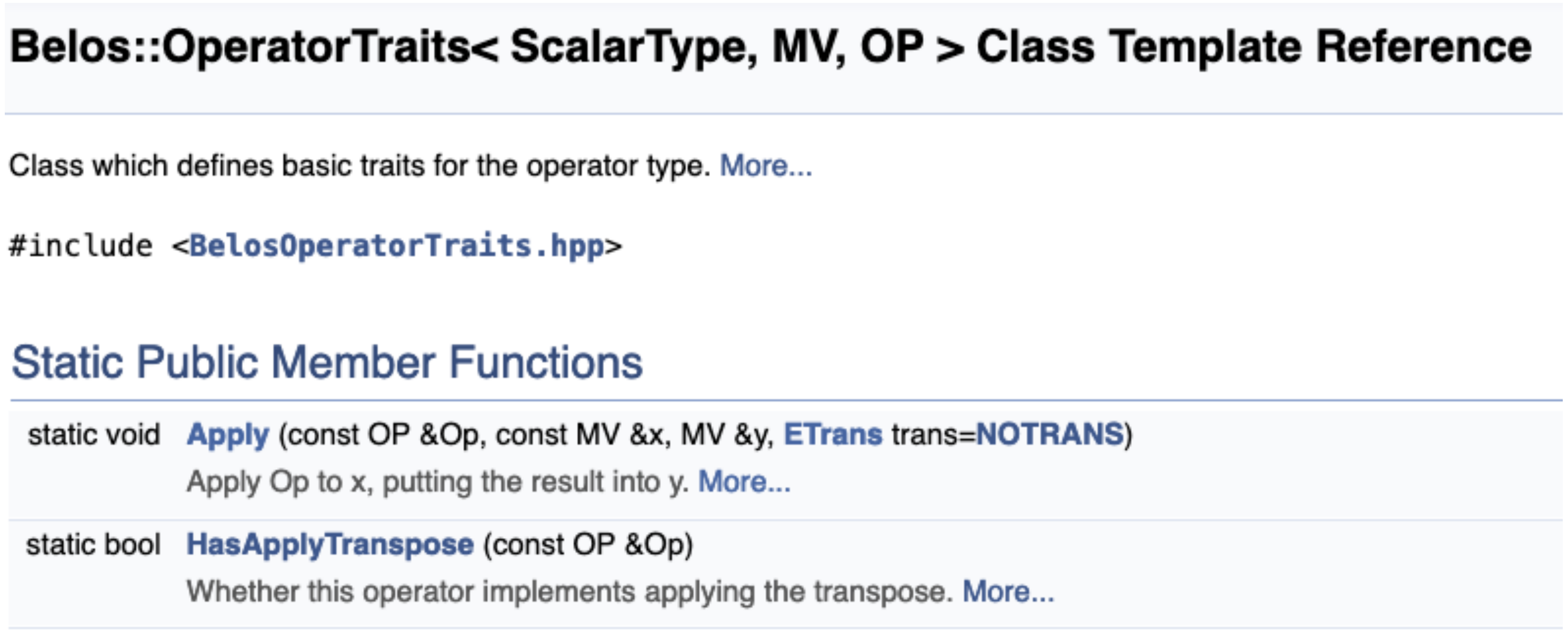
- Yes, use **Flexible GMRES**. (FGMRES)
- Possible use cases:
  - Adaptive preconditioner that changes at each Krylov iteration.
  - Low precision (e.g. FP32 or FP16) preconditioner.
  - Solver to precondition another solver. (Yes, you can use GMRES to precondition FGMRES!)
- Implemented as a parameter for the BlockGMRES solver: (not available for PseudoBlock)

```
belosList.set( "Flexible Gmres", true );
```

*Note: There is no reason to turn on the “Flexible” GMRES option if you are using a constant preconditioner! It will only increase costs.

What about matrix-free operators and preconditioners?

- Belos fully supports matrix-free operators and preconditioners!
- Simply implement Belos::OperatorTraits for your operator and selected multivector type.
- Polynomial preconditioning (based upon GMRES polynomial) available for general nonsymmetric and/or matrix-free operators. See GmresPolySolMgr.


Belos::OperatorTraits< ScalarType, MV, OP > Class Template Reference

Class which defines basic traits for the operator type. More...

```
#include <BelosOperatorTraits.hpp>
```

Static Public Member Functions

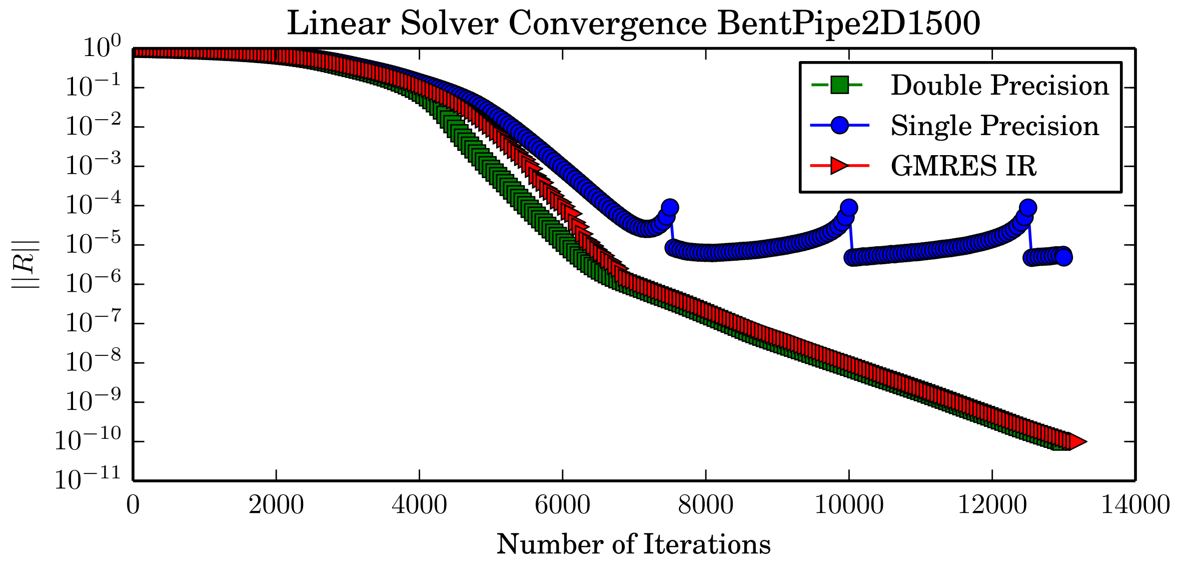
static void	Apply (const OP &Op, const MV &x, MV &y, ETrans trans=NOTRANS) Apply Op to x, putting the result into y. More...
static bool	HasApplyTranspose (const OP &Op) Whether this operator implements applying the transpose. More...

https://docs.trilinos.org/dev/packages/belos/doc/html/classBelos_1_1OperatorTraits.html

Belos: Communication-Avoiding and Pipelined Solvers

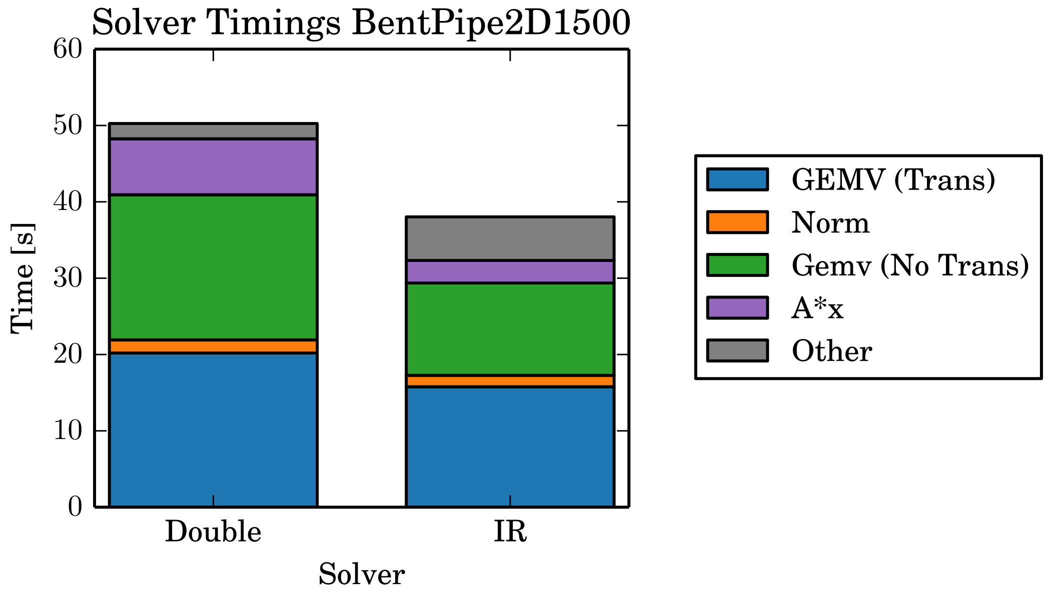
- A **communication-avoiding** solver has modifications to the original algorithm to reduce MPI communication (sends & receives) and/or overlap it with other operations (pipelining). Communication avoiding algorithms may also avoid local memory access (e.g., replacing BLAS2 with BLAS3), so the orthogonalization can run faster with one MPI rank.
- **TSQR** – Orthogonalization option (for GMRES and GCRO-DR) that is communication-avoiding. (Available for Epetra and Tpetra. Requires extra CMake options to enable.)
- **Single-Reduce CG** option available in Belos for both BlockCG and PseudoBlockCG when solving one RHS only.
- New **Tpetra-only** Belos solvers:
  - Located in Trilinos/packages/belos/tpetra/src/solvers.
  - Main author: Ichi Yamazaki (iyamaza@sandia.gov)
  - Enabled in Belos::SolverFactory. (Soon to be available through Stratimikos!)
- Available **communication-avoiding Tpetra-only** solvers (all take only one RHS):
  - Pipelined CG
  - Single-Reduce CG
  - Pipelined GMRES
  - S-Step GMRES [includes impl of CholQR]
  - Single-Reduce GMRES (a.k.a. “one-synch”)

Coming Soon: GMRES-IR (Mixed Precision GMRES)



- 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!

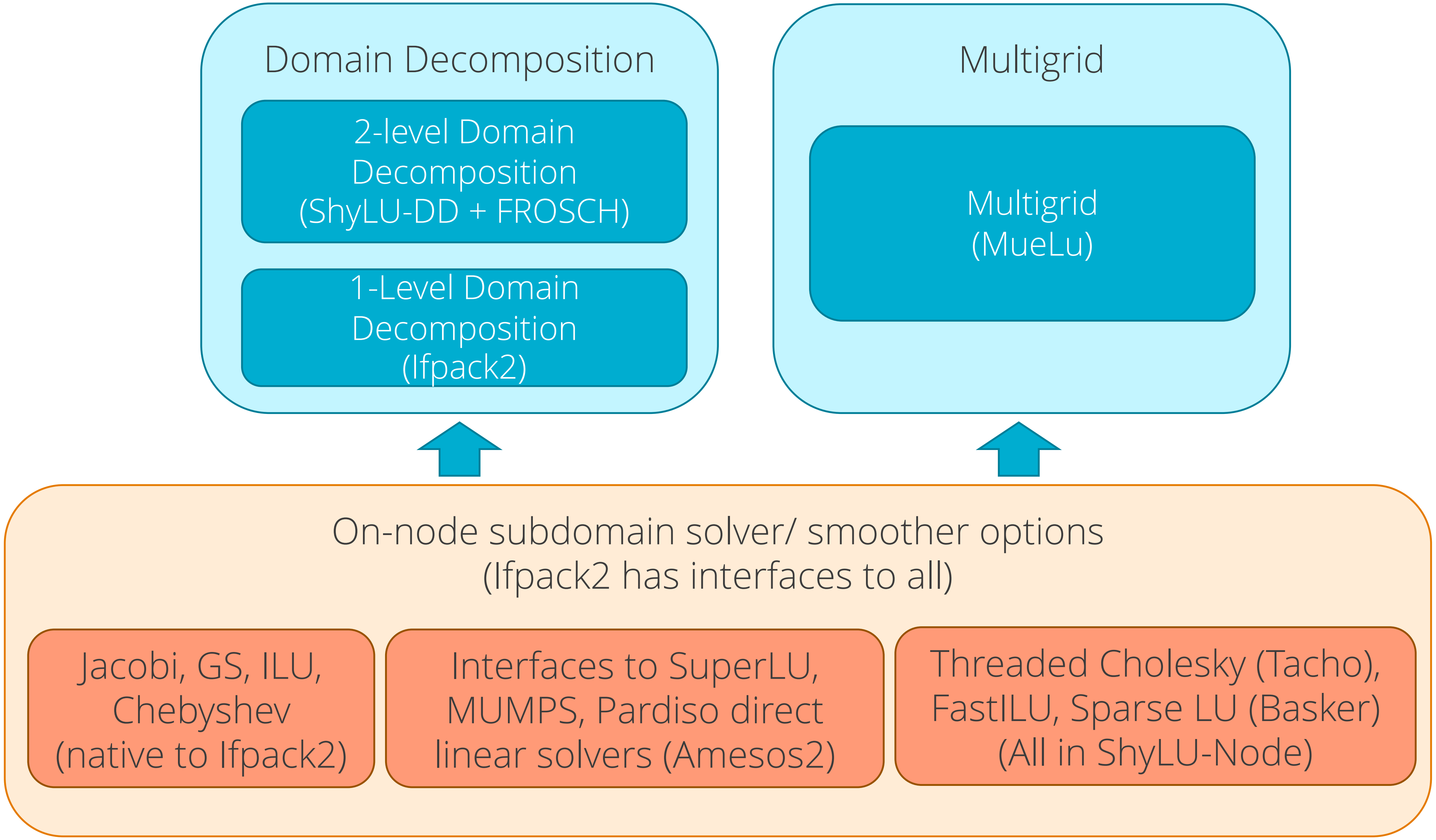
- 32% speedup for this example over all-double precision.
- Requires storing two copies of A (or implementing two operators).



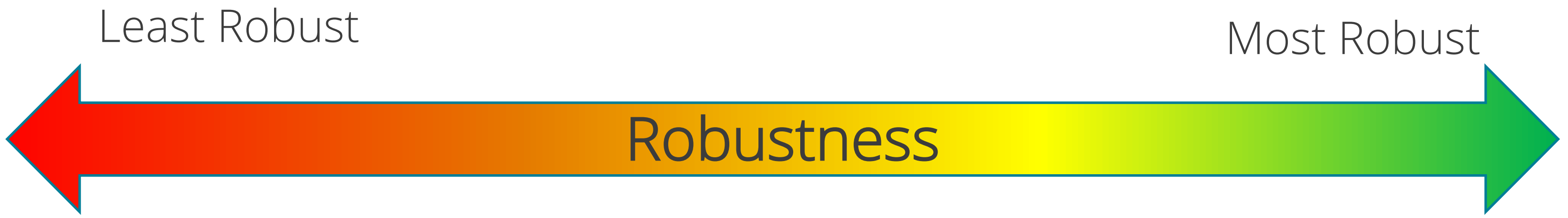
Belos/Krylov Solvers Resources and References:

- Amesos2 and Belos: Direct and iterative solvers for sparse linear systems [link]
  Eric Bavier, Mark Hoemmen, Sivasankaran Rajamanickam, Heidi Thornquist
- Belos Doxygen: https://docs.trilinos.org/dev/packages/belos/doc/html/index.html
- A Comparison of Preconditioned Krylov Subspace Methods for Large-Scale Nonsymmetric Linear Systems [link]. Aditi Ghai, Cao Lu and Xiangmin Jiao
- Examples in Trilinos/packages/belos/epetra/tests/*
  and in Trilinos/packages/belos/tpetra/tests/*

Overview of Preconditioning in Trilinos:


Domain Decomposition
2-level Domain Decomposition (ShyLU-DD + FROSCH)
1-Level Domain Decomposition (Ifpack2)
Multigrid
Multigrid (MueLu)
On-node subdomain solver/ smoother options (Ifpack2 has interfaces to all)
Jacobi, GS, ILU, Chebyshev (native to Ifpack2)
Interfaces to SuperLU, MUMPS, Pardiso direct linear solvers (Amesos2)
Threaded Cholesky (Tacho), FastILU, Sparse LU (Basker) (All in ShyLU-Node)

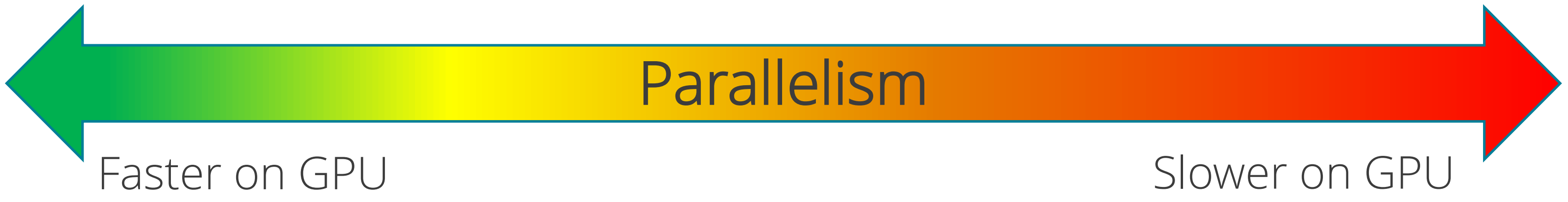
Incomplete LU factorization (ILU) Preconditioning Choices for Thermal Fluids Application:



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



*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 RUIL(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)

Trilinos Preconditioning Resources and References:

- FastILU: Finegrained ASynchronous iterative ILU. [Slides here]
- ShyLU: A Collection of Node-Scalable Sparse Linear Solvers [slides here]
- ShyLU-FROSCH: https://shylu-frosch.github.io/about/
- Ifpack2 User's Guide [here]
- Trilinos Framework and Solvers [slides here]
- Trilinos 10.10 tutorial [here]

Amesos2

- Amesos2 is a direct solver package (interfaces to many direct solvers).
- Three native solvers: KLU2 (default), Basker, ShyLU/Tacho

- Required packages: Teuchos, Tpetra, Kokkos
- Optional packages: Epetra, ShyLU
- Optional TPLs: MPI, SuperLU, MUMPS, Pardiso, (Par)METIS, LAPACK, Strumpack

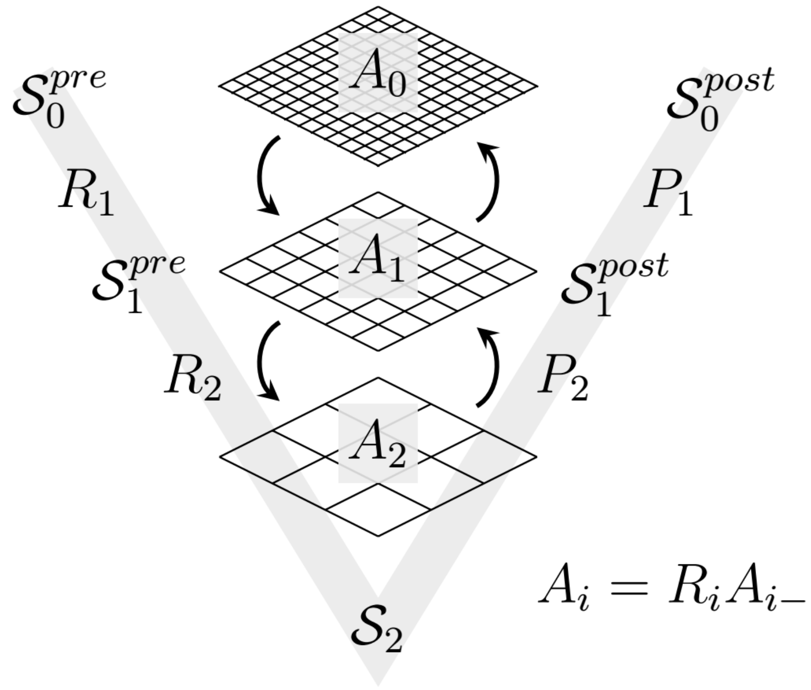
- Capabilities: numeric factorizations (KLU), and more...

Amesos2: Recent Developments

- Supernodal SpTRSV for SuperLU/Cholmod
  - Kokkos-based: runs SpTRSV on GPU
  - Utilizes “supernodal” block structures (block operations, hierarchical parallelism).
  - Implements several algorithms including “partitioned inverse.”
    - Transforms SpTRSV into a sequence of SpMVs (one SpMV / level).
- ShyLU/Tacho
  - Kokkos-based: runs on CUDA/AMD GPU
  - Level-set based sparse factorization (not tasking; “original” Tacho = Task-parallel sparse Cholesky)
  - Supports LDLT, LU, Cholesky (symmetric indefinite, symmetric sparsity pattern, SPD, respectively)
- ShuLU/Basker
  - “Threaded” version of KLU2
  - Threaded-performance has been improved in recent years.
  - Primarily used for circuit design (e.g., Xyce).
- Amesos2 (SolverCore) now implements iterative refinement
  - Beneficial for recovering potential accuracy degradation due to limited pivoting (e.g., in parallel factorization).

MueLu

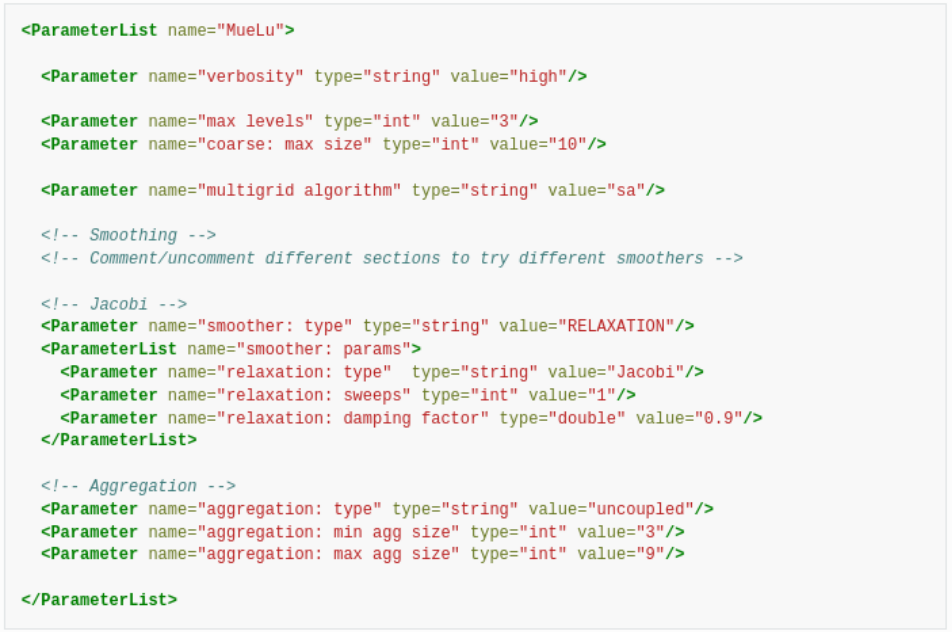
- MueLu is a multigrid solving/preconditioning package.
- Part of the second-generation of Trilinos
  - Templated on scalars, ordinals, and nodes
- Multigrid is an optimal complexity O(n) solver for linear systems.
  1. Start with a "fine grid"
  2. Smooth error, transfer to coarser grid
  3. Repeat 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)

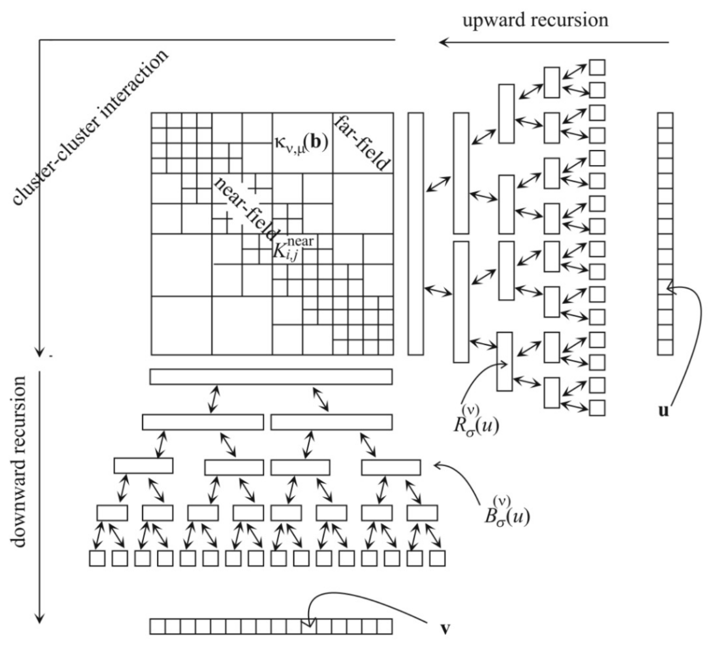


```xml
<ParameterList name="MueLu">

  <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

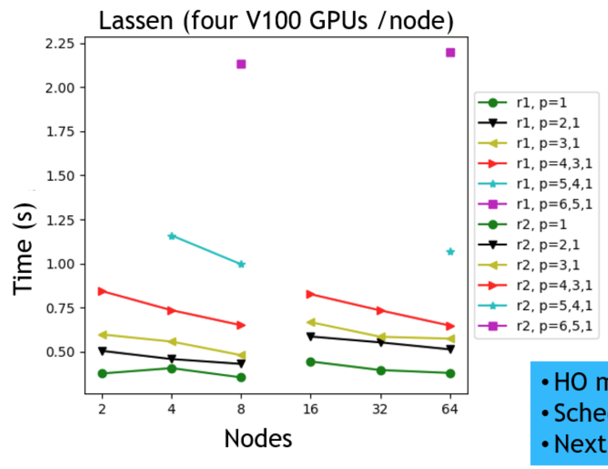
MueLu Developments

- Hierarchical Matrices
  - For applications with dense matrices
- Maxwell1
  - Multigrid for electromagnetics
- Multiprecision
  - Do coarse levels 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.

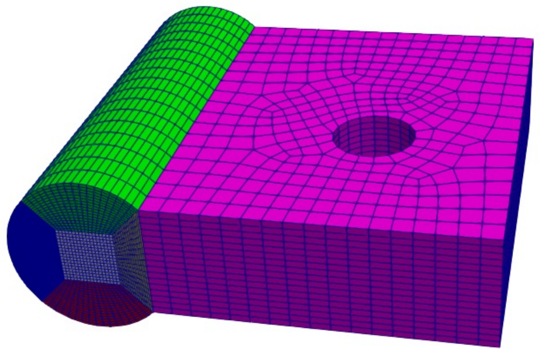


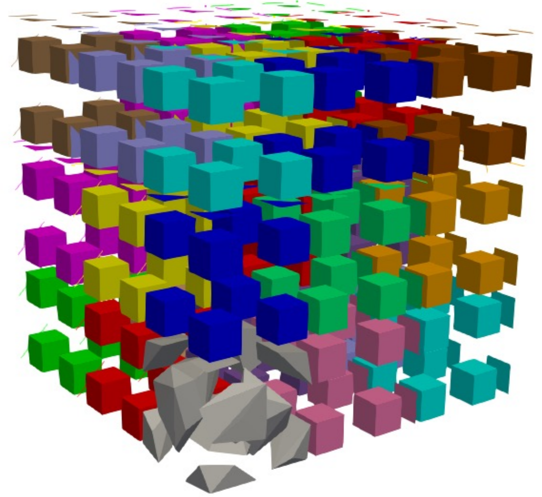
p coarsening schedule	nnz/row	Iterations	
		R1	R2
1	16	30	31
2 -> 1	43	24	24
3 -> 1	84	27	26
4 -> 3 -> 1	144	21	20
5 -> 4 -> 1	224	32	32
6 -> 5 -> 1	328	63	63

- 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

- Region Multigrid
  - Grids of grids
  - Subgrids coarsened geometrically or algebraically
- Geometric Multigrid
  - Problems with structured meshes
- NotayAggregation
  - Pairwise aggregation
- Cut Drop
  - Tackles fill-in from Sa-AMG
  - For dropping weak distant connections





Classical AMG SoC: A_{ij} is strong if,

$$-A_{ij} \geq \theta \max_{k \neq i} -A_{ik},$$

SA SoC: A_{ij} is strong if,

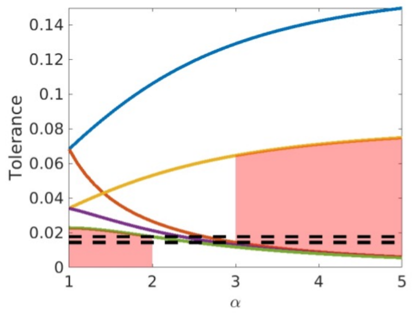
$$|A_{ij}| \geq \theta \sqrt{A_{ii} A_{jj}},$$

Distance Laplacian: A_{ij} is strong if:

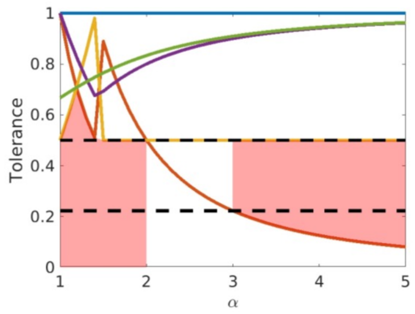
$$|L_{ij}| \geq \theta \sqrt{L_{ii} L_{jj}},$$

where: $$L_{ij} = -\|\mathbf{x}_i - \mathbf{x}_i\|^{-2}, L_{ii} = -\sum_{i \neq j} L_{ij}.$$

Distance Laplacian

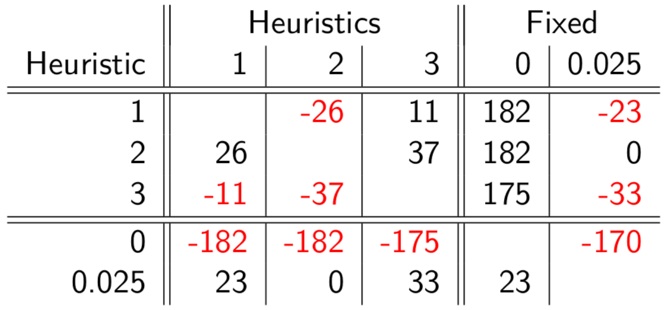


Cutting

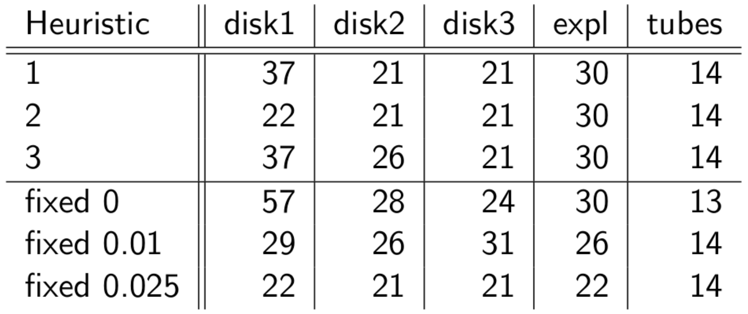


Upcoming MueLu Developments & More

- 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	



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

- BlockCRS
  - Preserve (small) fixed blocked structure
- Matrix-Free
  - MF tentative prolongator operator
  - Hierarchy treats R,P as operators
  - MF -> matrix coarse operator next
  - Synergistic with high order work and Maxwell

Thank you!

Questions?

Contacts:

Belos: Jennifer Loe jloe@sandia.gov

MueLu: Graham Harper gbharpe@sandia.gov

Communication-Avoiding Belos Solvers: Ichi Yamazaki iyamaza@sandia.gov

Ifpack2: Jonathan Hu jhu@sandia.gov

ShyLU / Amesos2: Siva Rajamanickam srajama@sandia.gov