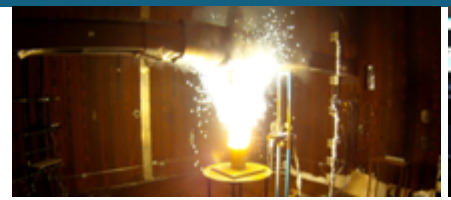
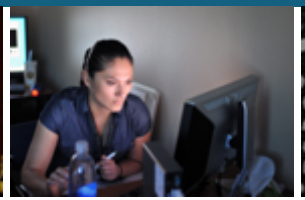
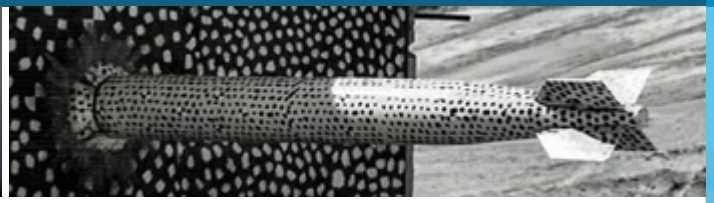
Experimental Evaluation of Multiprecision Strategies for GMRES on GPUs


Jennifer Loe, Christian Glusa, Ichitaro Yamazaki, Erik Boman, Siva Rajamanickam

Sandia National Laboratories

EXASCALE COMPUTING PROJECT

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

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.

SAND2021-5975 C

The idea behind GMRES the (Generalized Minimum RESidual Method):

To solve $Ax = b$, where A is $n \times n$:

1. Build an orthonormal basis for a Krylov subspace:

$$\text{span}\{b, Ab, A^2b, \ldots, A^{m-1}b\}$$

2. Use an orthogonal projection to find an approximate solution which minimizes the residual:

$$\| b - Ax \|_2$$

GMRES (Generalized Minimum RESidual) Algorithm:

Algorithm GMRES (Modified Gram-Schmidt)

1: $\gamma = \|b\|_2$ and $v_1 = b/\gamma$
2: **for** $j = 1 : m$ **do**
3: $\quad w_j = Av_j$
4: $\quad$ **for** $i = 1 : j$ **do**
5: $\quad\quad h_{ij} = v_i^T w_j$
6: $\quad\quad w_j = w_j - h_{ij} v_i$
7: $\quad$ **end for**
8: $\quad h_{j+1,j} = \|w_j\|_2.$
9: $\quad v_{j+1} = w_j / h_{j+1,j}$
10: **end for**
11: Define the $(m+1) \times m$ matrix $\overline{H} = \{h_{ij}\}$
12: Solve least-squares problem $\overline{H}d = \gamma e_1$ for d.
13: $\hat{x} = V_m d$

Sparse Matrix-Vector Product (SpMV)

Orthogonalizing the next basis vector

Restart when subspace size gets too large!

See details in “Iterative Methods for Sparse Linear Systems 2nd ed.” by Saad.

Why incorporate lower precisions in GMRES?

- Reduce data movement to overcome memory-bound algorithms.
- Use cheaper floating-point operations.

Obstacles to lower precision:

- Lower precision computations result in more roundoff error!
- …but applications still need high level of accuracy in solutions.
- Tricky to find where to use lower precision in algorithm while maintaining accuracy.

So how DO we use lower precision in GMRES?

Iterative Refinement with GMRES (GMRES-IR)

Algorithm 1 Iterative Refinement with GMRES Error Correction

1: $r_0 = b - Ax_0$ [double]
2: **for** $i = 1, 2, \ldots$ until convergence: **do**
3: Use GMRES(m) to solve $Au_i = r_i$ for correction u_i [single]
4: $x_{i+1} = x_i + u_i$ [double]
5: $r_{i+1} = b - Ax_{i+1}$ [double]
6: **end for**

(At each restart, update solution vector and recompute residuals in double precision.)

Note: We store TWO copies of matrix A (double and single).

Not a new algorithm. See related works:

- Neil Lindquist, Piotr Luszczek, and Jack Dongarra. *Improving the performance of the GMRES method using mixed-precision techniques.*
- Hartwig Anzt, Vincent Heuveline, and Bjorn Rocker. *Mixed precision iterative refinement methods for linear systems: Convergence analysis based on Krylov subspace methods.*
- Erin Carson and Nicholas J. Higham. *Accelerating the solution of linear systems by iterative refinement in three precisions.*

Implementation of Krylov Solvers in Trilinos

- Belos: Linear Solvers package in Trilinos:
  - All linear algebra kernels are abstracted through "adapter" interface.
  - Solvers interface does not support mixing precisions! Mixed precision must occur through the adapter.
- Kokkos and Kokkos Kernels:
  - Portable parallel linear algebra.
  - Performant BLAS kernels for GPU (single node).
- New Mixed Precision Krylov Solvers Software:
  - New adapter to use Kokkos as the linear algebra backend for solvers.
  - Tested performance on a single node with V100 GPU.

Experiment Setup:

Algorithm 1 Iterative Refinement with GMRES Error Correction

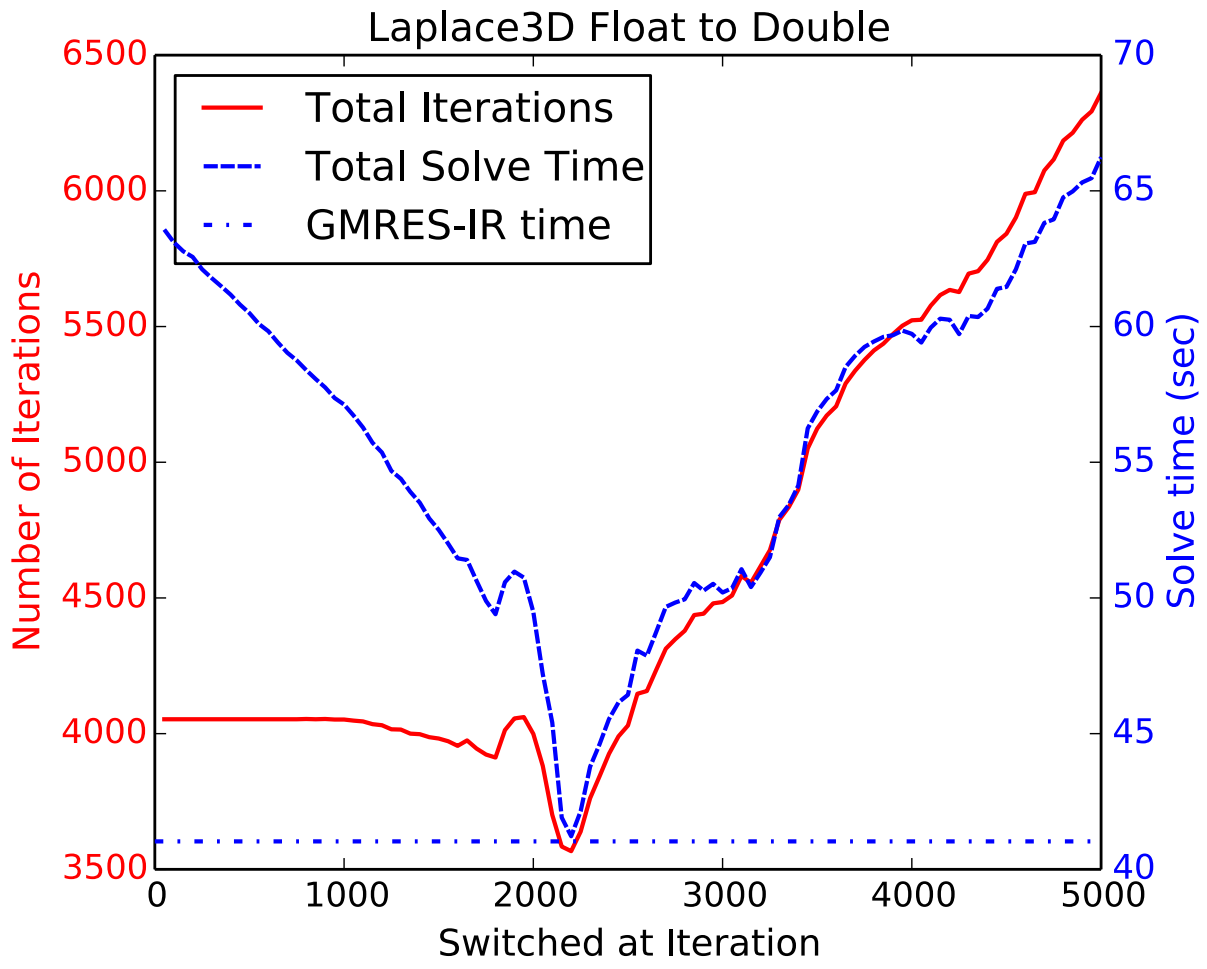
1: $r_0 = b - Ax_0$ [double]
2: **for** $i = 1, 2, \ldots$ until convergence: **do**
3: Use GMRES(m) to solve $Au_i = r_i$ for correction u_i [single]
4: $x_{i+1} = x_i + u_i$ [double]
5: $r_{i+1} = b - Ax_{i+1}$ [double]
6: **end for**

Experiment parameters:

- Restarting GMRES at every 50 iterations.
- Recompute residuals in double at each restart (step 4 & 5).
- Stopping when relative residual less than 1e-10. ($|| b - Ax||_2 / ||b|| < 10^{-10}$)
- Tests run on a V100 GPU.

GMRES-IR wins over "switching" strategy (GMRES-FD):

- What if we run GMRES in single precision and then switch to double precision?
- But where to switch?
- Matrix: 3D Laplacian, nx=200
- GMRES-FD (float-double switch)
  - Min solve time: 41.22s
  - Min iterations: 3567
- GMRES-IR:
  - Solve time: 41.03s
  - Iterations: 4100
- **GMRES-IR attains the same minimum solve time as the switching strategy! No need to choose a switching point!**



How does convergence of GMRES-IR compare to GMRES double?

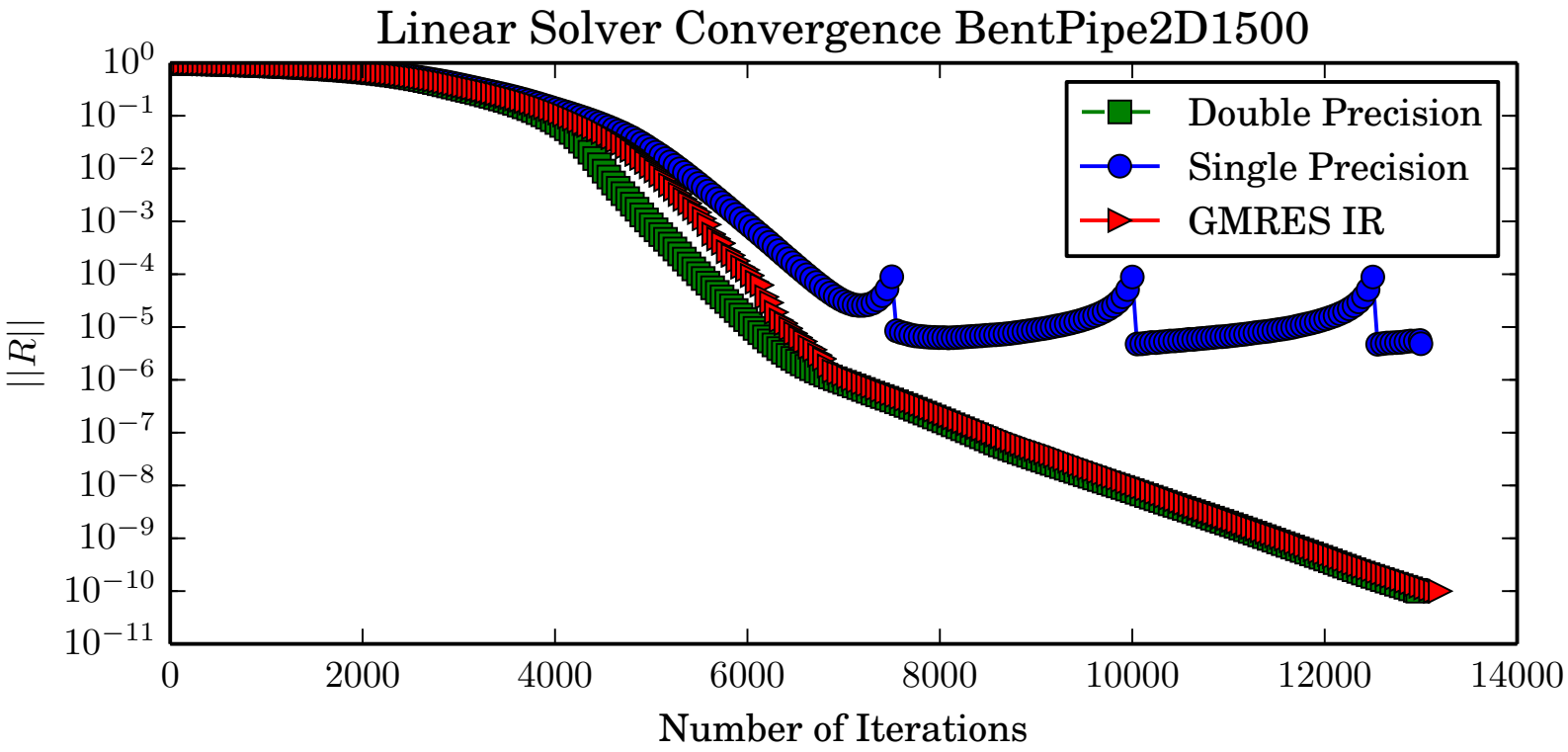
Test Problem:

- 2D convection-diffusion
- 5-pt stencil
- (Highly nonsymmetric.)
- n = 2.25 million
- nnz = 11,244,000
- Convergence Tolerance: 1e-10

GMRES Convergence:
Single: Stalls near 1e-5

Double: 12,967 iterations
50.26 seconds

IR: 13,150 iterations
38.03 seconds



GMRES-IR convergence follows convergence of GMRES Double!

Kernel Speedup

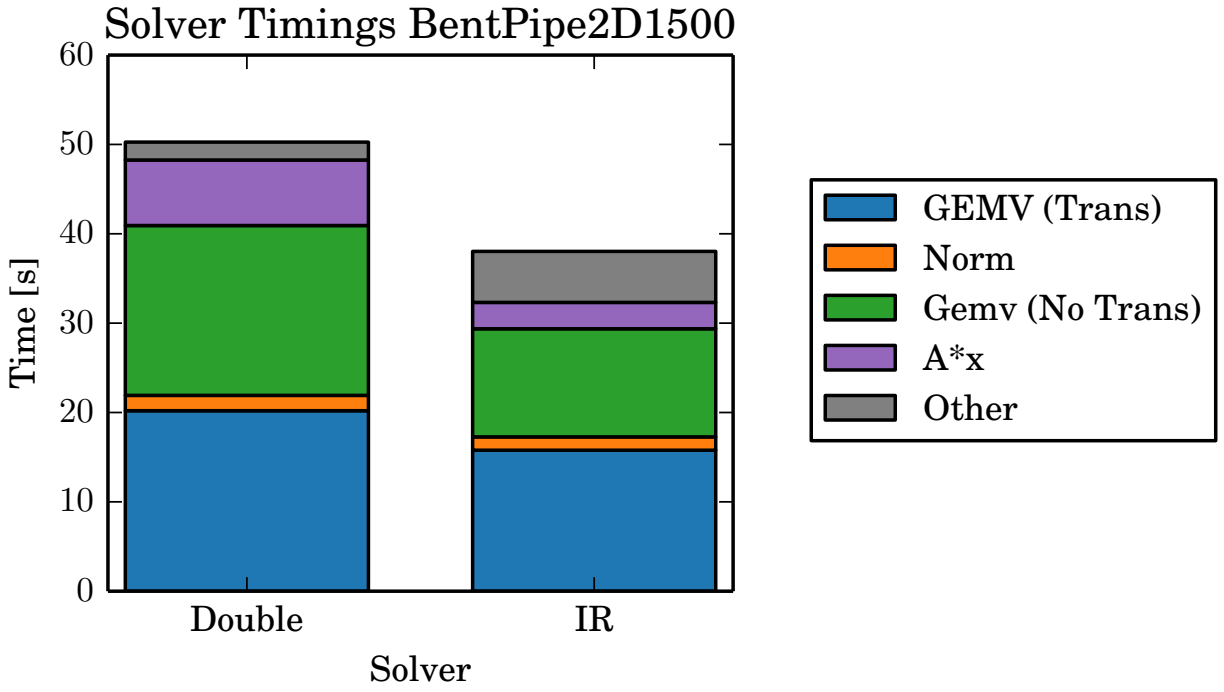
Problem information:
(Same as previous slide.)

- 2D convection-diffusion problem
- n = 2.25 million

GMRES Double converges in:
12,967 iterations
50.26 seconds

GMRES-IR converges in:
13,150 iterations
38.03 seconds
(3.98 sec on double precision operations)

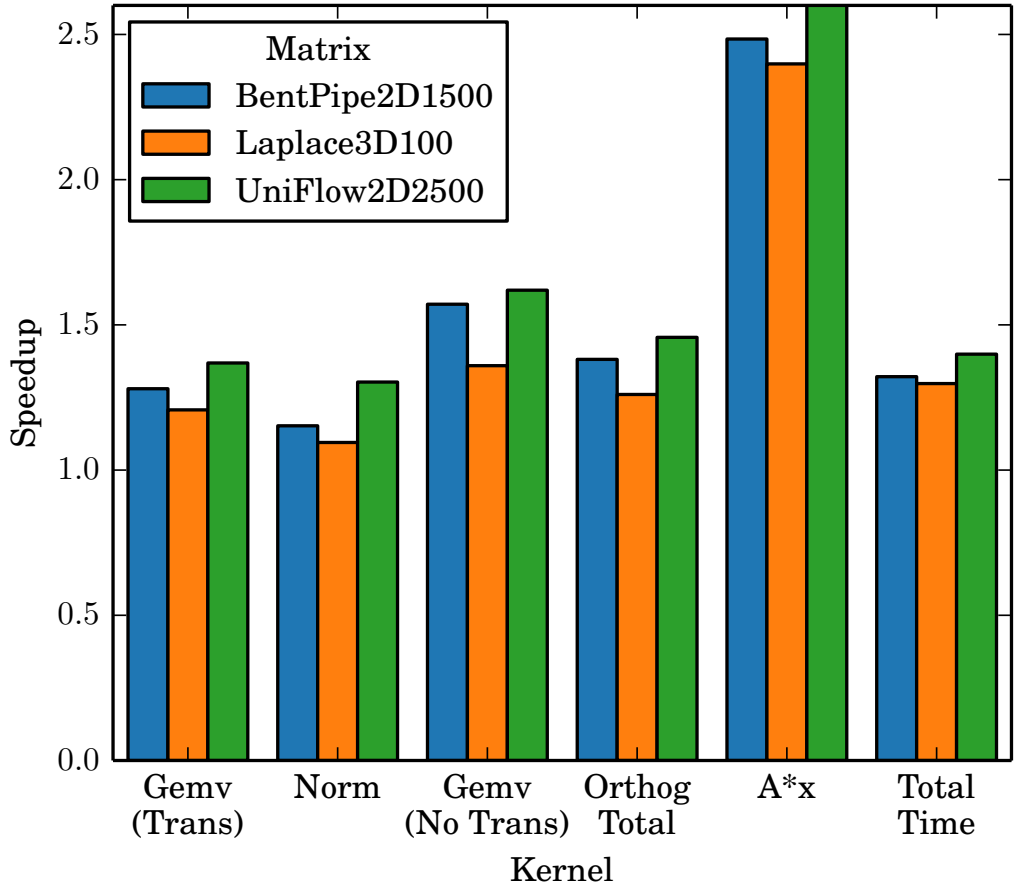
- (Timings do not include making an extra copy of the matrix A in single precision.)
- Using 2 steps of Classical Gram-Schmidt orthogonalization for each iteration (CGS2)



	GMRES double	GMRES IR	Speedup
Total time:	**50.26**	**38.03**	**1.322**
Ortho: GEMV Trans	20.20	15.78	1.280
Ortho: GEMV No Trans	19.01	12.10	1.571
Ortho (norm)	1.71	1.49	1.152
A*x	7.33	2.95	2.484

Kernel speedups compared with other matrices:

- Matrices:
  - Bent Pipe (previous 2 slides)
  - 3D Laplacian
  - 2D UniFlow convection-diffusion
- Same kernels presented as previous slide.
- SpMV with A*x gets 2.4 to 2.6 times speedup in float precision. Why??
- NVIDIA Profiling tools showed increased L2 cache hit rate for SpMV in float as compared to double.
- fp 32 gets "perfect caching" for x, while fp 64 has to reread elements of x into cache.



Total of 1.3x to 1.4x speedup over 3 different PDE problems.

A model for L2 cache use with low precision SpMV:

Suppose that A has w nonzero elements per row and n rows (so $nnz = w*n$).

A stored in CSR format with 2 vectors of size $w * n$:

Values of A: A_{val} Column indices: $colId$ (Ignore vector of row ptrs)

Computing the first dot product of the SPMV:

$$\sum_{i=0}^{w-1} A_{val}[i] * x[colId[i]].$$

Case: fp64 with no cache reuse (i.e. every element of x has to be read into cache every time needed):

$$n * w * [size(int) + 2 * size(double)] = 20wn.$$

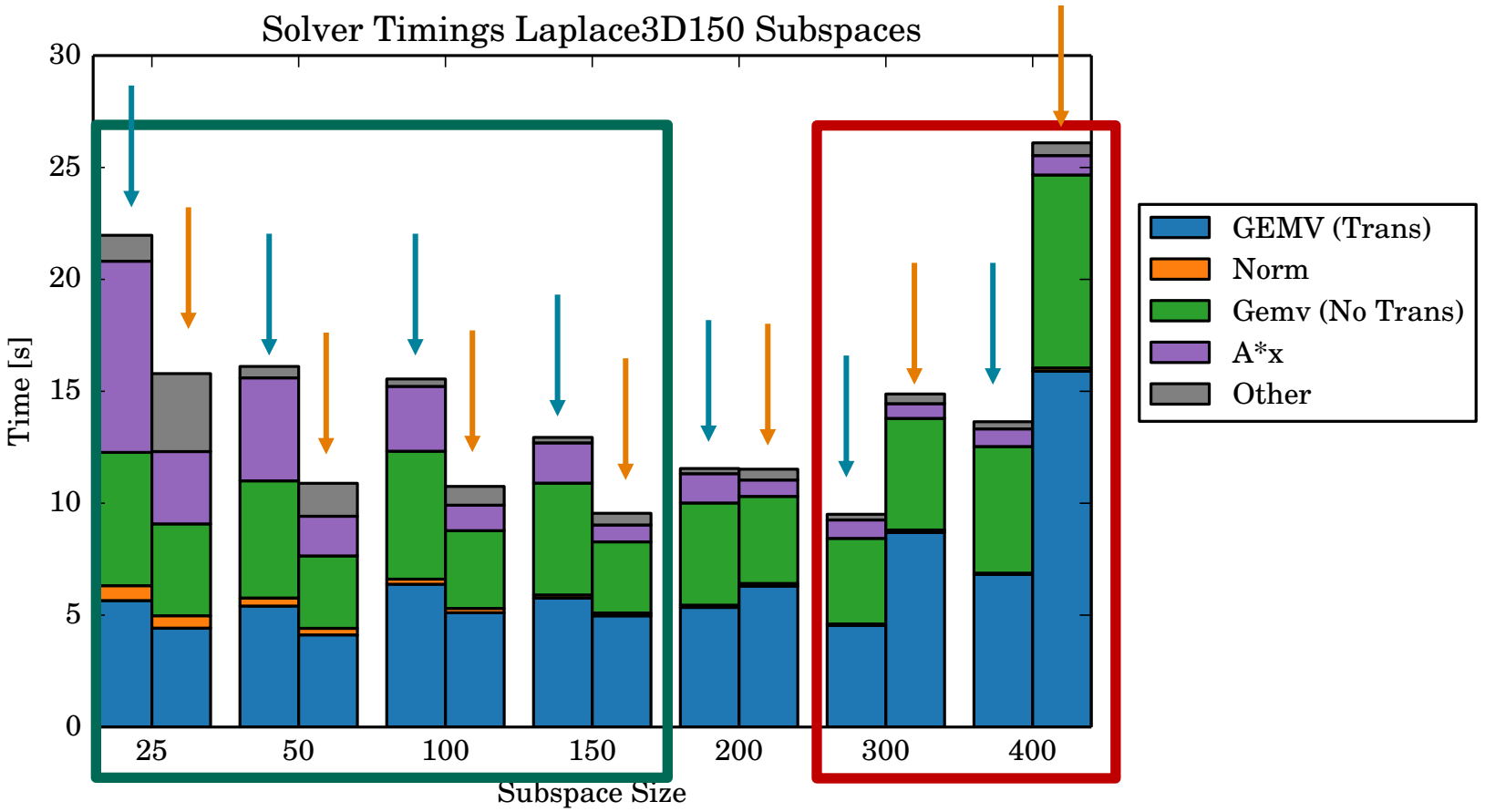
Case: fp32 with "perfect" cache reuse (i.e. any elements of x read into cache stay in cache until not needed):

$$n*w*[size(int)+size(float)]+n*size(float) = (8w+4)n.$$

Expected speedup: $\frac{20wn}{(8w+4)n} = \frac{5w}{2w+1}.$ → 2.5 as w gets large.

** Thanks to Christian Trott and Luc-Berger Vergiat for help in creating this model!

How does the Krylov subspace (restart) size affect solve time?



Small Subspace:
GMRES-IR fastest

Large Subspace:
(Less practical in reality.)
GMRES double fastest

Problem:

- 3D Laplacian, nx=150
- Different restart sizes. (x-axis)

- GMRES Double: Left bars
- GMRES-IR: Right bars

Takeaway: GMRES-IR gives speedup with small-medium restart lengths.
However, with large restarts, it needs extra time to converge.

How does preconditioning affect GMRES-IR convergence?

Test Problem:

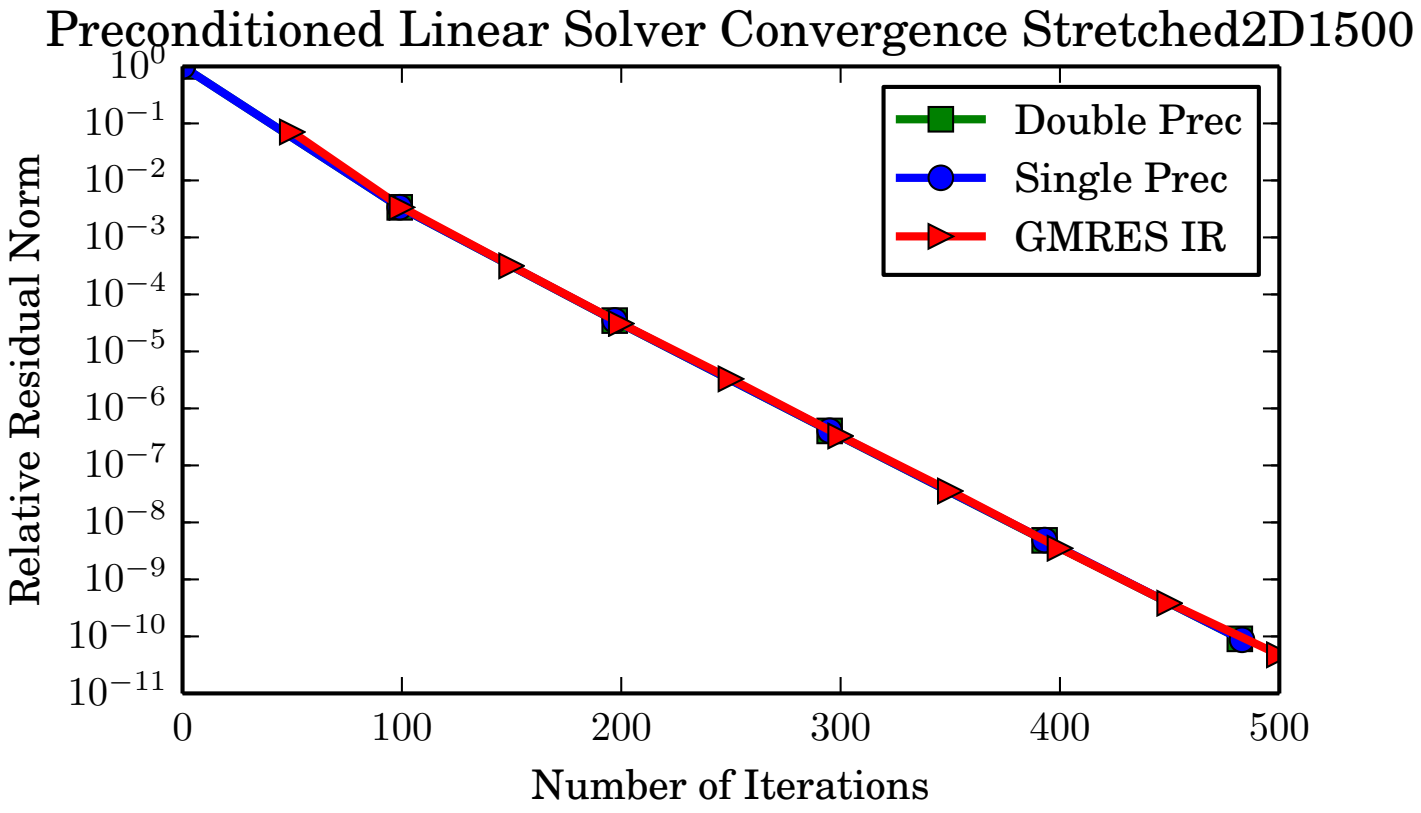
- 2D Laplacian
- Stretched Grid
- n = 2.25 million
- Convergence Tolerance: 1e-10

Preconditioning:

- Polynomial preconditioner
- Right preconditioning:

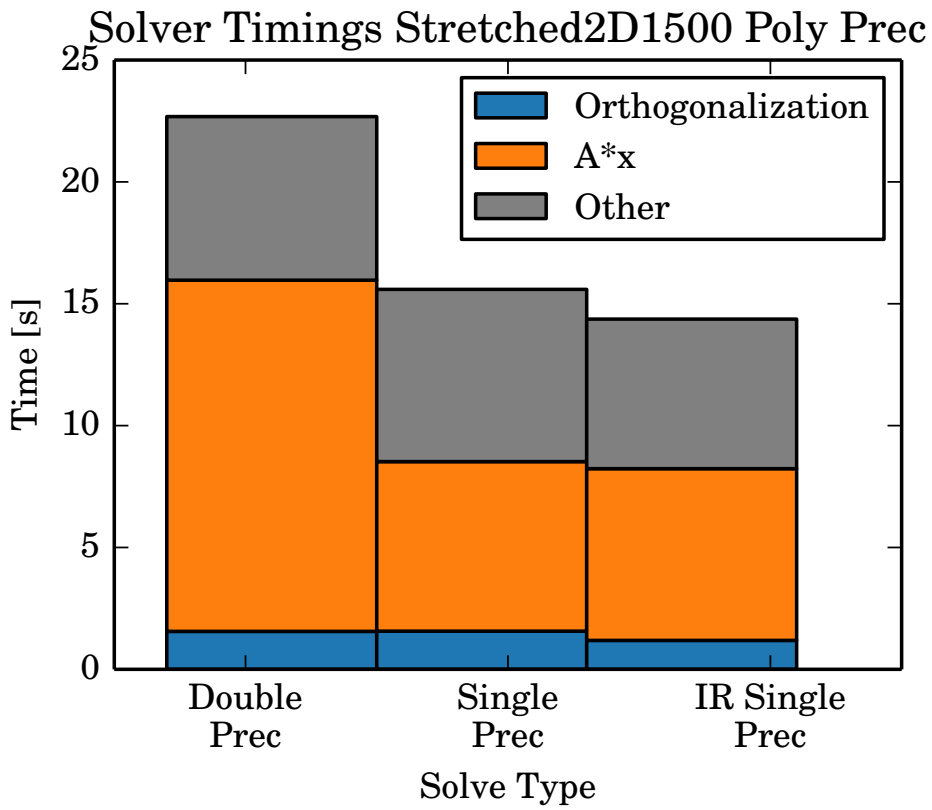
$$AMM^{-1}x = b$$

- Polynomial based upon GMRES
- GMRES double has double precision polynomial.
- GMRES-IR has single precision polynomial.



Preconditioned GMRES-IR convergence still follows convergence of GMRES Double!

Polynomial Preconditioning



Test Problem (same as previous slide):

- 2D Laplacian, Stretched Grid
- n = 2.25 million
- Polynomial preconditioner based upon GMRES polynomial**

Three solves compared:

1. GMRES double w/ double precision polynomial. (left)
2. GMRES double w/ single precision polynomial. (middle)
3. GMRES-IR w/ single precision polynomial. (right)

- (Solve times do not include preconditioner creation.)

Takeaways:

- Single precision preconditioning improves solve time ~ 30%.
- GMRES-IR improves solve time even more.
- Polynomial preconditioning shifts main expense to SpMV rather than dense orthogonalization kernels.

**For polynomial preconditioning details, see:
Jennifer Loe, Erik Boman, and Heidi Thornquist.
Polynomial Preconditioned GMRES in Trilinos: Practical Considerations for High-Performance Computing

Results from SuiteSparse Matrices:

			Double		IR		
UF id	Matrix Name	N	Time	Iters	Time	Iters	Speedup
2266	atmosdmodj	1,270,432	5.12	1740	3.78	1750	1.35
1849	Dubcova3	146,698	1.15	1131	1.05	1150	1.10
895	stomach	213,360	0.51	359	0.52	400	0.98
1367	SiO2	155,331	18.23	17385	16.86	17600	1.08
1853	parabolic_fem	525,825	41.77	27493	45.34	36600	0.92
805	cfd2*	123,440	6.05	1092	4.55	1100	1.33
2649	Transport*	1,602,111	8.35	339	8.73	450	0.96
1431	filter3D*	106,437	25.24	4449	18.12	4450	1.39
	BentPipe2D1500	2,250,000	50.26	12967	38.03	13150	1.32
	UniFlow2D2500	6,250,000	29.62	2905	21.17	3000	1.40
	Laplace3D150	3,375,000	16.93	2387	11.75	2400	1.44

Example PDE stencil problems from previous slides.

* with degree 25 polynomial preconditioning

Results from SuiteSparse Matrices:

			Double		IR		
UF id	Matrix Name	N	Time	Iters	Time	Iters	Speedup
2266	atmosdmodj	1,270,432	5.12	1740	3.78	1750	1.35
1849	Dubcova3	146,698	1.15	1131	1.05	1150	1.10
895	stomach	213,360	0.51	359	0.52	400	0.98
1367	SiO2	155,331	18.23	17385	16.86	17600	1.08
1853	parabolic_fem	525,825	41.77	27493	45.34	36600	0.92
805	cfd2*	123,440	6.05	1092	4.55	1100	1.33
2649	Transport*	1,602,111	8.35	339	8.73	450	0.96
1431	filter3D*	106,437	25.24	4449	18.12	4450	1.39
	BentPipe2D1500	2,250,000	50.26	12967	38.03	13150	1.32
	UniFlow2D2500	6,250,000	29.62	2905	21.17	3000	1.40
	Laplace3D150	3,375,000	16.93	2387	11.75	2400	1.44

Quickly converging problems; not much room for speedup from GMRES-IR.

* with degree 25 polynomial preconditioning

Results from SuiteSparse Matrices:

			Double		IR		
UF id	Matrix Name	N	Time	Iters	Time	Iters	Speedup
2266	atmosdmodj	1,270,432	5.12	1740	3.78	1750	1.35
1849	Dubcova3	146,698	1.15	1131	1.05	1150	1.10
895	stomach	213,360	0.51	359	0.52	400	0.98
1367	SiO2	155,331	18.23	17385	16.86	17600	1.08
1853	parabolic_fem	525,825	41.77	27493	45.34	36600	0.92
805	cfd2*	123,440	6.05	1092	4.55	1100	1.33
2649	Transport*	1,602,111	8.35	339	8.73	450	0.96
1431	filter3D*	106,437	25.24	4449	18.12	4450	1.39
	BentPipe2D1500	2,250,000	50.26	12967	38.03	13150	1.32
	UniFlow2D2500	6,250,000	29.62	2905	21.17	3000	1.40
	Laplace3D150	3,375,000	16.93	2387	11.75	2400	1.44

Different from other results. Why does double precision convergence not follow GMRES-IR convergence for this problem??

* with degree 25 polynomial preconditioning

Results from SuiteSparse Matrices:

			Double		IR		
UF id	Matrix Name	N	Time	Iters	Time	Iters	Speedup
2266	atmosdmodj	1,270,432	5.12	1740	3.78	1750	1.35
1849	Dubcova3	146,698	1.15	1131	1.05	1150	1.10
895	stomach	213,360	0.51	359	0.52	400	0.98
1367	SiO2	155,331	18.23	17385	16.86	17600	1.08
1853	parabolic_fem	525,825	41.77	27493	45.34	36600	0.92
805	cfd2*	123,440	6.05	1092	4.55	1100	1.33
2649	Transport*	1,602,111	8.35	339	8.73	450	0.96
1431	filter3D*	106,437	25.24	4449	18.12	4450	1.39
	BentPipe2D1500	2,250,000	50.26	12967	38.03	13150	1.32
	UniFlow2D2500	6,250,000	29.62	2905	21.17	3000	1.40
	Laplace3D150	3,375,000	16.93	2387	11.75	2400	1.44

Very good speedup for SuiteSparse test problems!

* with degree 25 polynomial preconditioning

Future Work:

- Implement GMRES-IR in Tpetra solvers in Belos package of Trilinos
- Make GMRES (double) with single precision preconditioning available in Tpetra Belos solvers.
- Test GMRES-IR on applications! (Ice sheet modeling)
- Incorporate half precision computations.
- Test performance on other (non-NVIDIA) GPU architectures- AMD and Intel.

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.