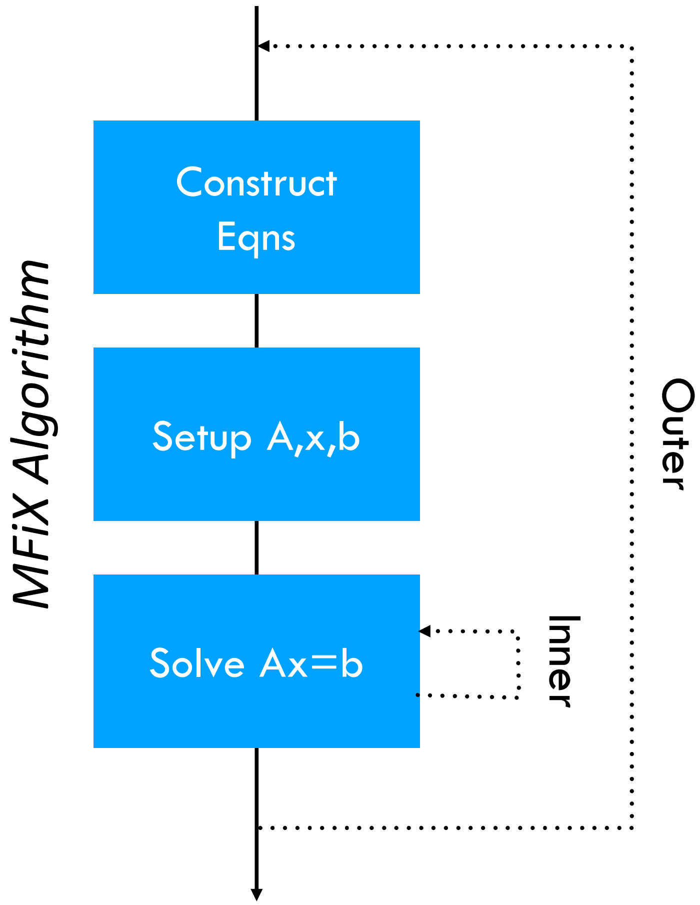
Interfacing MFIX with PETSC and HYPRE Linear Solver Libraries

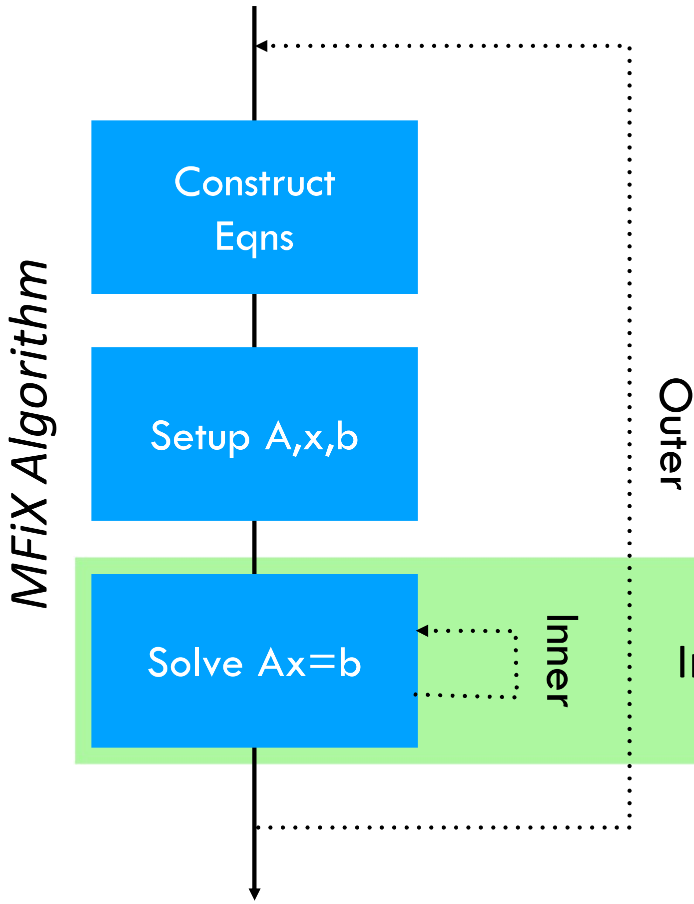
Jeremy Thornock, U. of Utah
Surya Yamujala, U. of Utah
Gautham Krishnamoorthy, U. of North Dakota
Lauren Clarke, U. of North Dakota

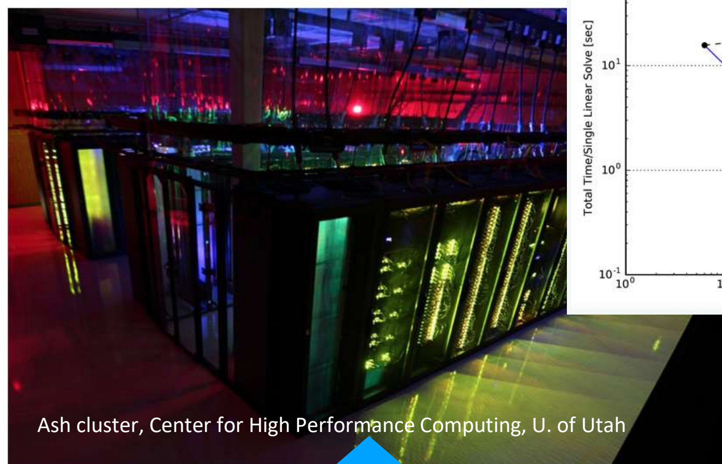
Acknowledgement: This research is being funded by the University Coal Research Program which is administered by DOE-NETL (Award #: DE-FE0026191) PM: Jason Hissam

Major Objective

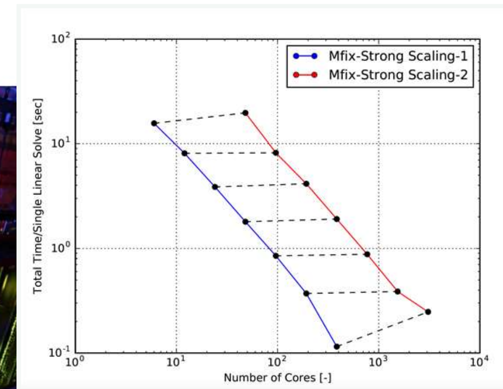


Problem Statement



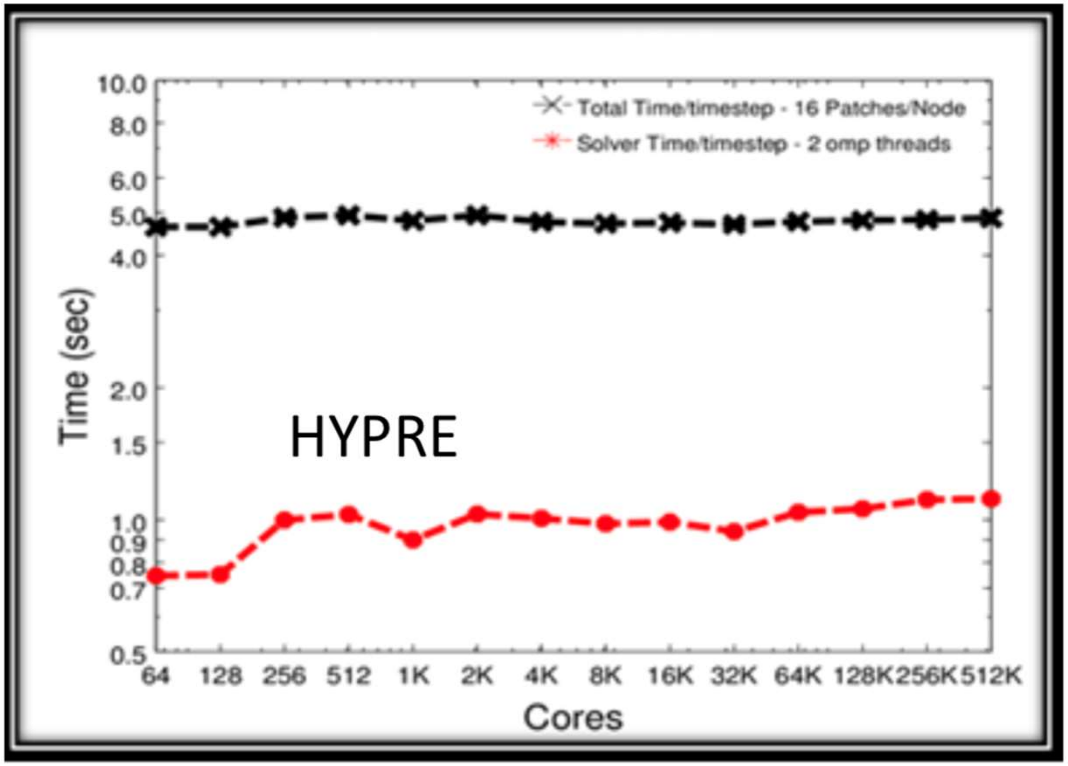


Ash cluster, Center for High Performance Computing, U. of Utah



Incorporate Hypre/PETSc as an inner solver option

Linear Solver Packages: Hypre and PETSc



- Both PETSc (ANL) and Hypre (LLNL) are fairly mature with active development
- Large user bases
- Many examples of good scaling up to large numbers of cores for sparse linear systems
- Both are C/C++ based codes with Fortran interfaces
- Roadmaps for heterogenous architecture (GPU, OpenMP, …)

Major Objectives

- **Non-disruptive** implementation with tech transfer
- **Verified implementations** of the third-party linear solvers
- Demonstrate **parallel scaling** on local resources (up to 7K cores)
- Demonstrate **algorithmic scaling** (robustness)

Team

Project Manager
G. Krishnamoorthy

HYPRE

J. Thornock, U. of Utah PI

Surya Yamujala, Student

PETSC

G. Krishnamoorthy, UND PI

Lauren Clarke, Student

- Experience with HYPRE in an in-house LES code.
- Symmetric Pressure Poisson

- Experience with PETSC and HYPRE for solving the RTE.
- RTE is non-symmetric

MFIX TEAM (Jordan Musser, Jeff Dietiker)
Jason Hissam

Computational Implementation

Development Principles

- ☑ Non-disruptive interface to the linear solver options
- ☑ Useable

- PETSc and Hypre are easy to build!
- Linking 3p packages handled with modification of environment variables during configure
- Leverage the existing modules for input file parameters (Bools, Ints, Floats, etc)
- New Fortran modules hold the 3p solver interface
- Logic in solve_lin_eq.f direct the algorithm to the selected solver (eqn dependent)

Code Example

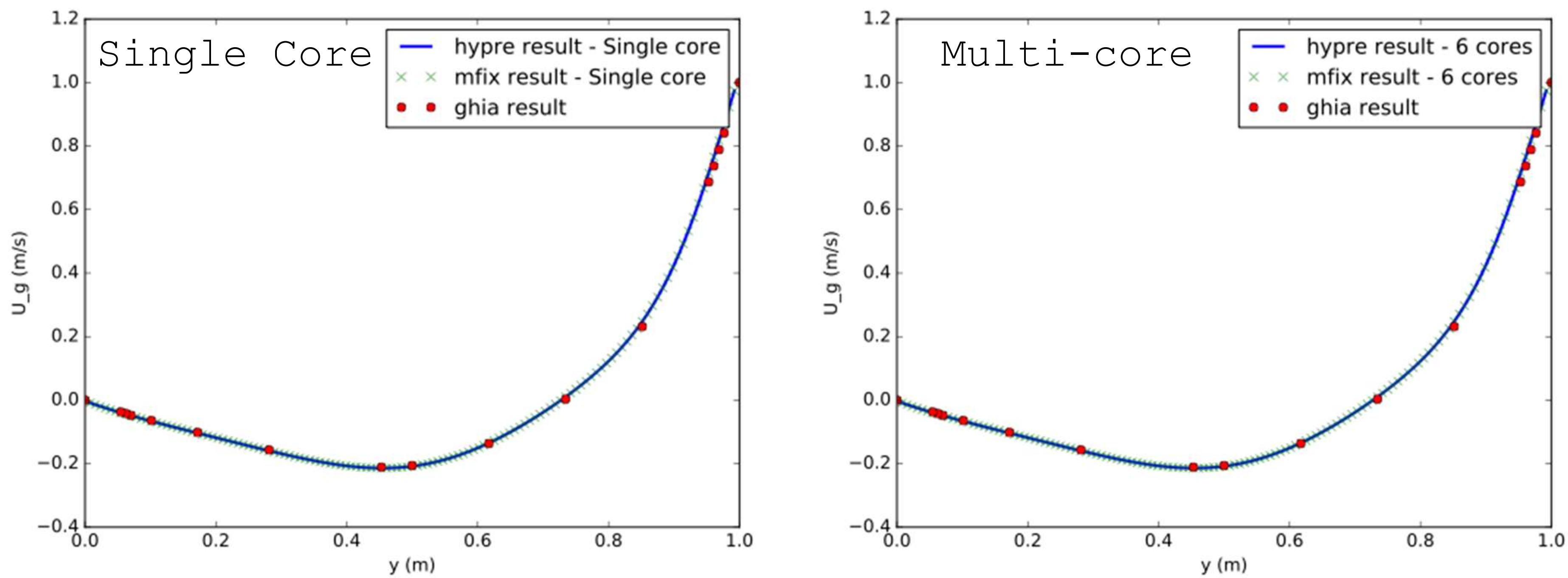
solve_lin_eq.f

```
!---------------------------------------------
! Turn off/on hypre solve:
     INTEGER :: DO_MPI_SETUP = 0
     LOGICAL :: DO_HYPRE_SOLVE = .False.

     CALL CHECK_FOR_HYPRE_SOLVE( VNAME, DO_HYPRE_SOLVE )

     IF ( DO_HYPRE_SOLVE .eqv. .TRUE. ) THEN

       CALL MPI_SETUP()

       CALL HYPRE_LIN_SOLVE( A_M, B_M, Var, &
                            DIMENSION_3, &
                            DIMENSION_M, &
                            M,           &
                            ISTART, IEND, &
                            JSTART, JEND, &
                            KSTART, KEND, &
                            DO_MPI_SETUP )

     ELSE
```

See: https://bitbucket.org/jthornock/mfix_hypre_integration for code and wiki documentation. Email J. Thornock for access.

Code Correctness

- Several cases have been tested to demonstrate correctness by comparing against known data or comparing to solution with the native MFiX solver.
- Tests have been performed on single and multiple cores



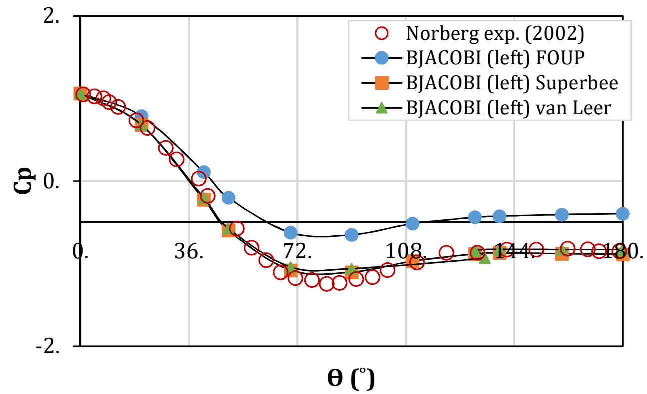
Lid driven cavity problem with momentum and pressure solves

Code Correctness

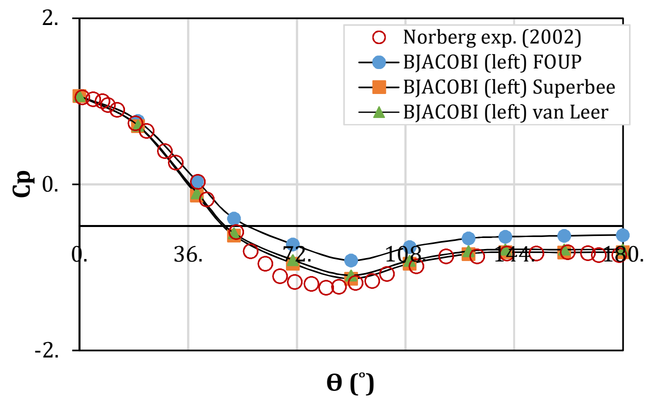
Accuracy

- Comparison of the pressure coefficient (Cp) results using left-side Block Jacobi preconditioning against experimental data
- Overall, higher-order discretization schemes compare better with experimental data compared to the lower order scheme (FOUP)
- There was little to no difference in results for all preconditioning methods that were tested for the coarse mesh and the intermediate mesh

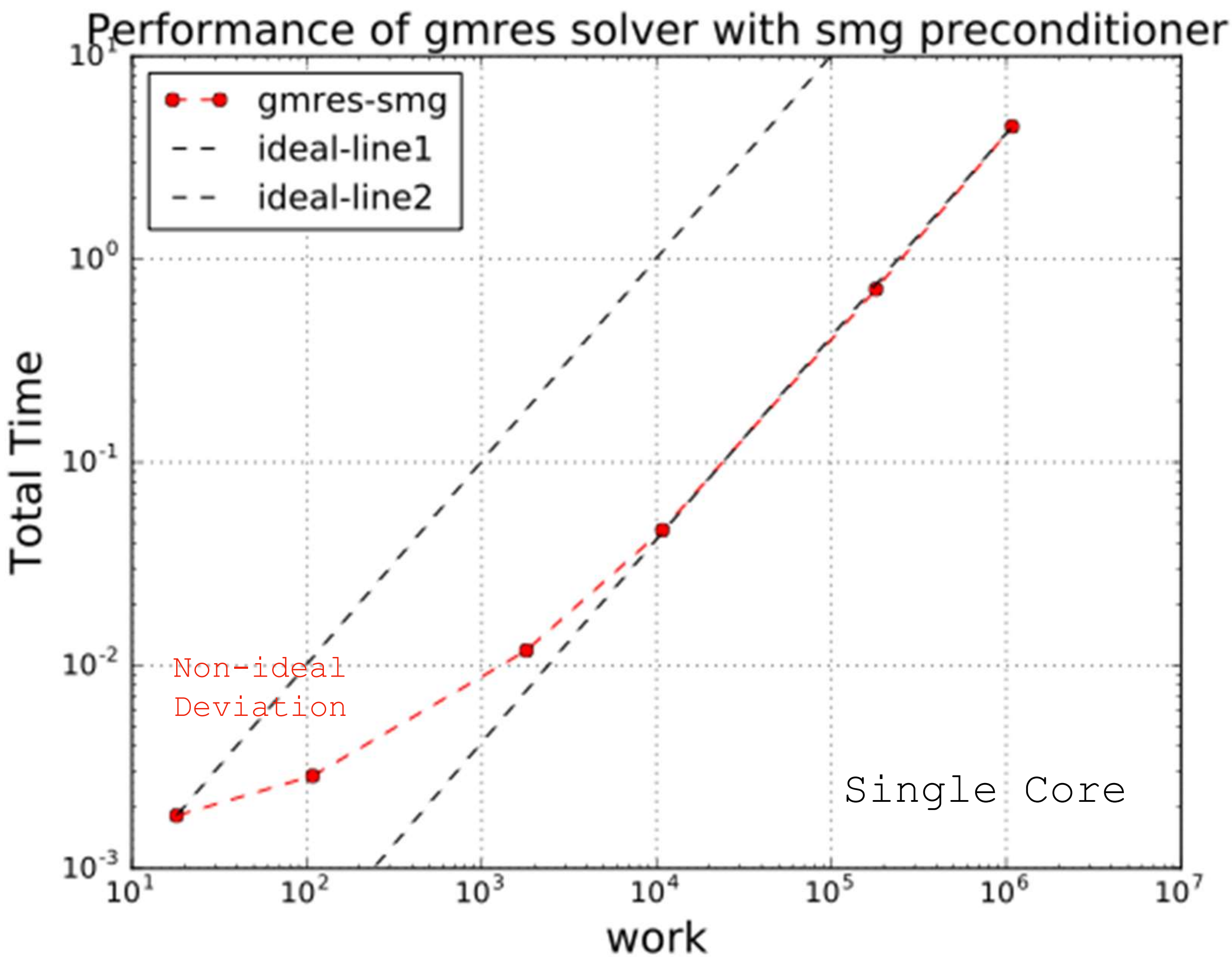
Coarse Mesh (120x80)



Intermediate Mesh (240x160)

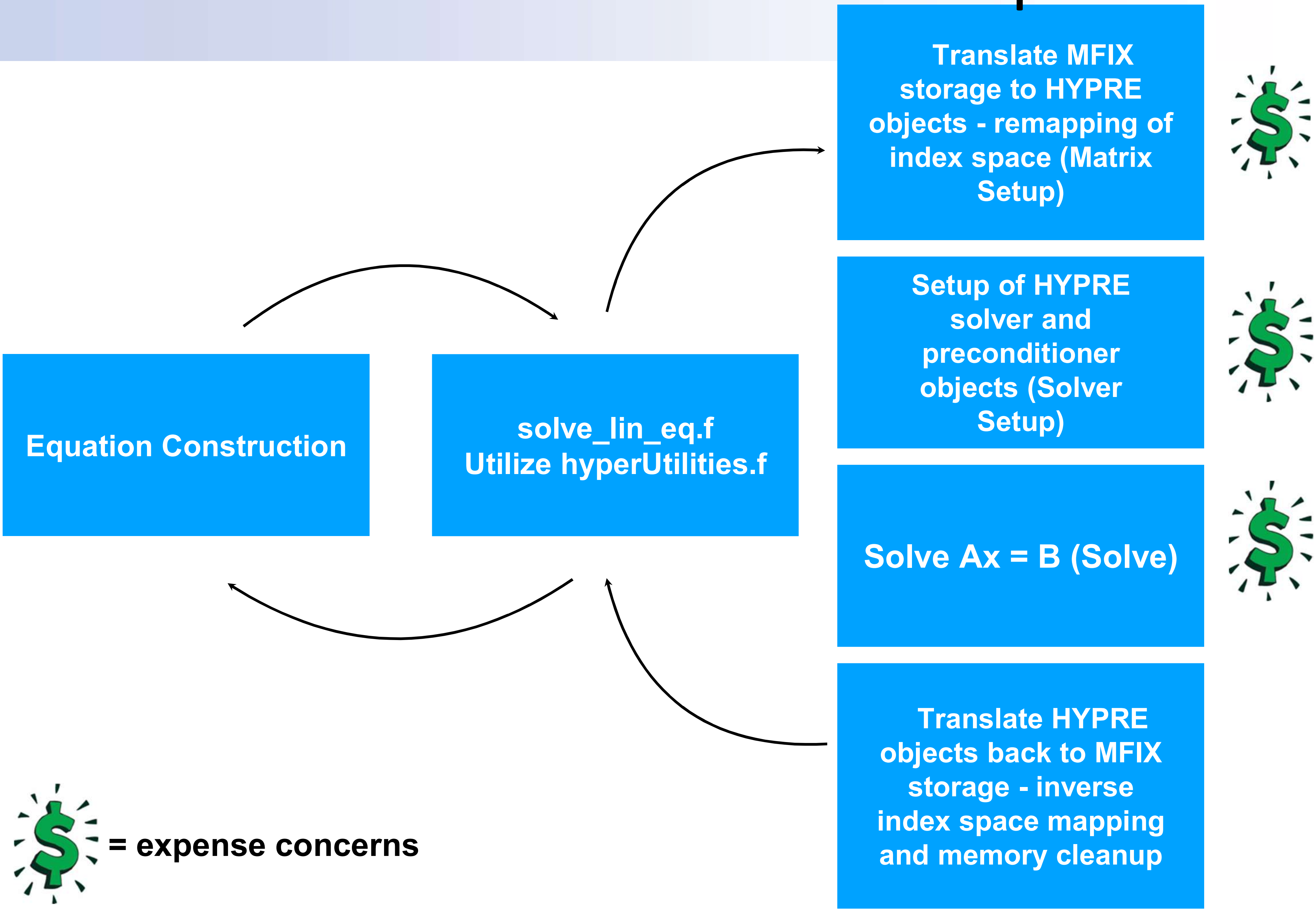


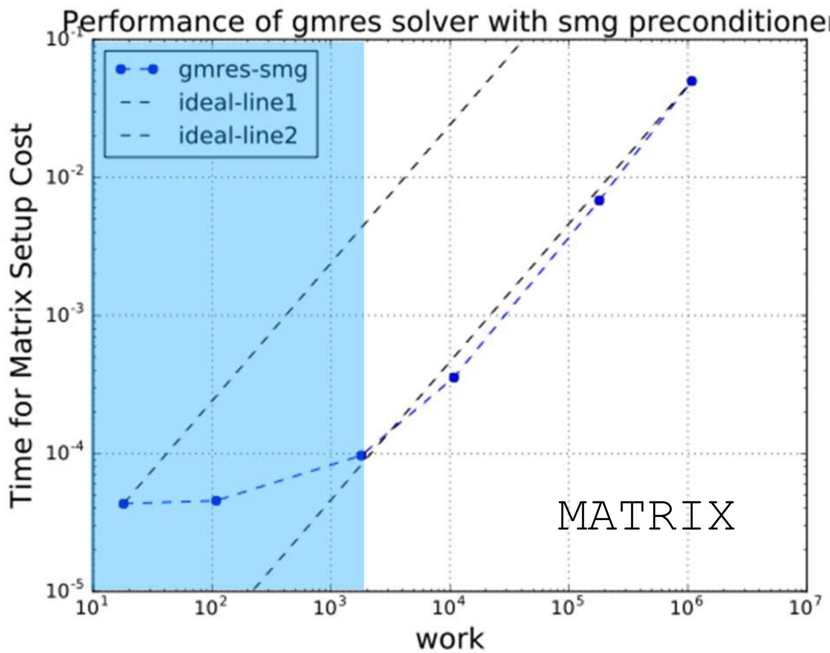
3P Overhead - Hypre

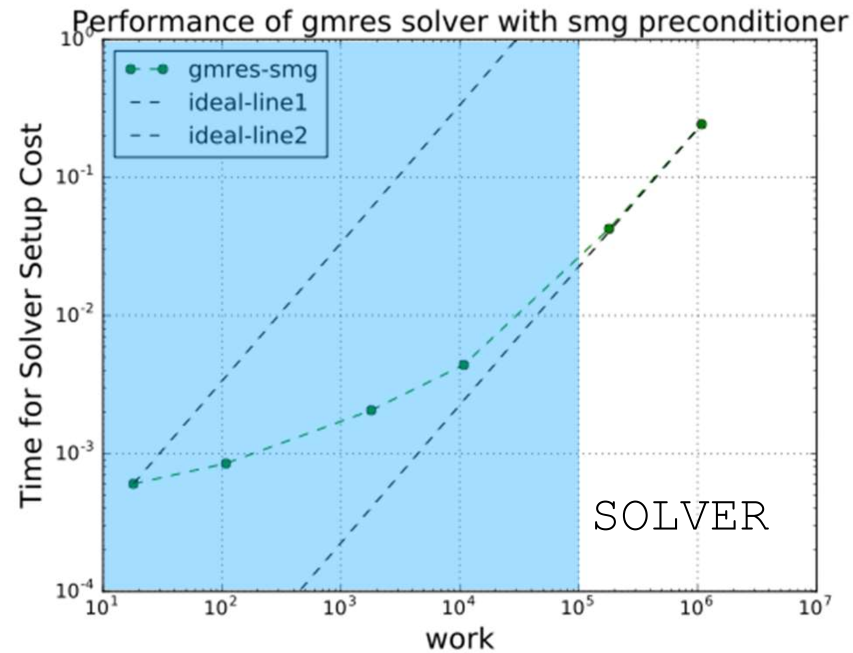


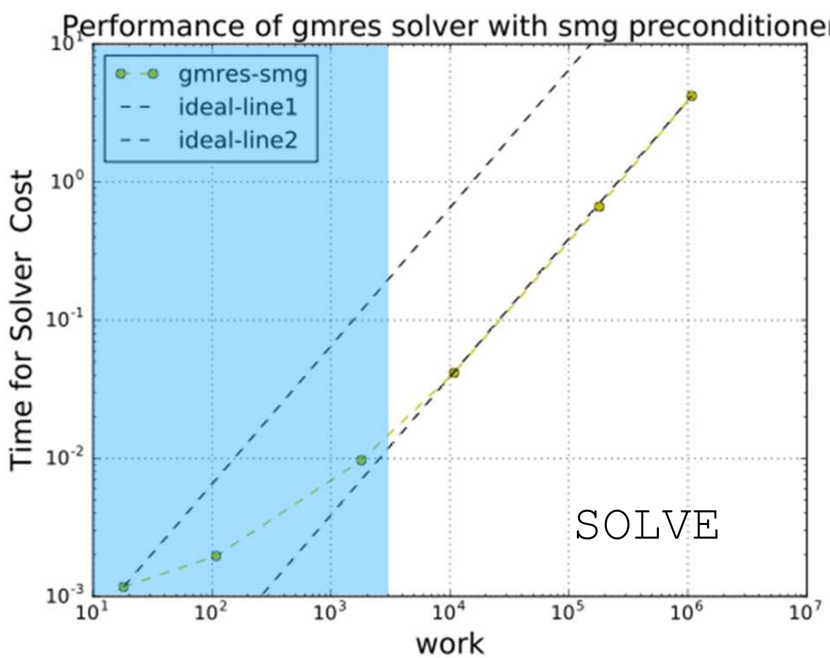
- 2-d heat transfer problem
- Work measured as total number of cells
- Ideal lines are just multiples of two from the left or right-most points

Solver Performance: Setup





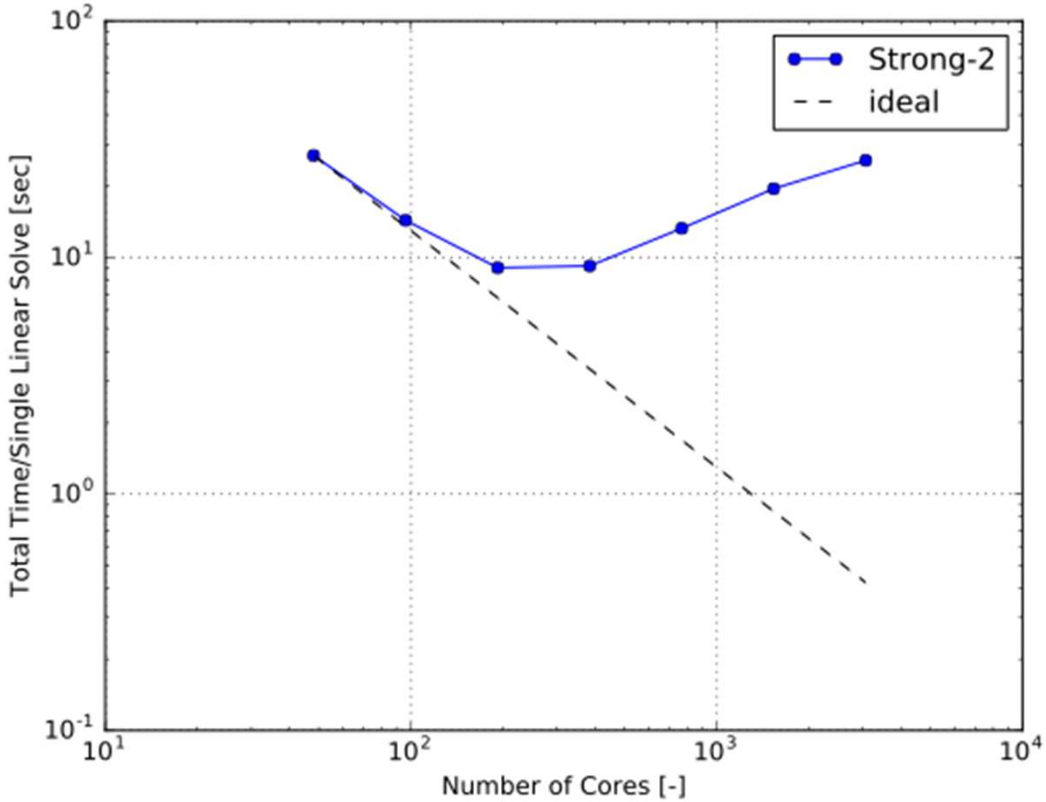


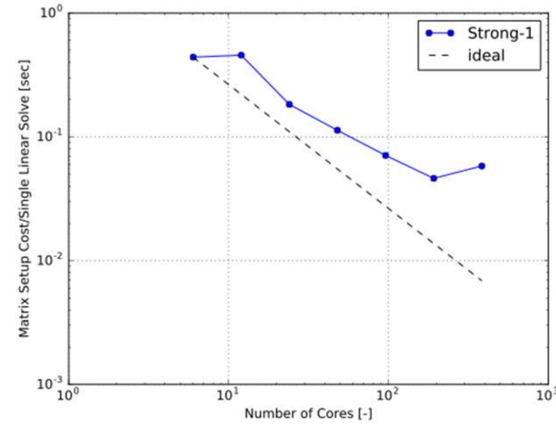


- Blue shaded regions exhibit non-ideal behavior
- Solver setup costs are particularly problematic
- Problem persists for multicores

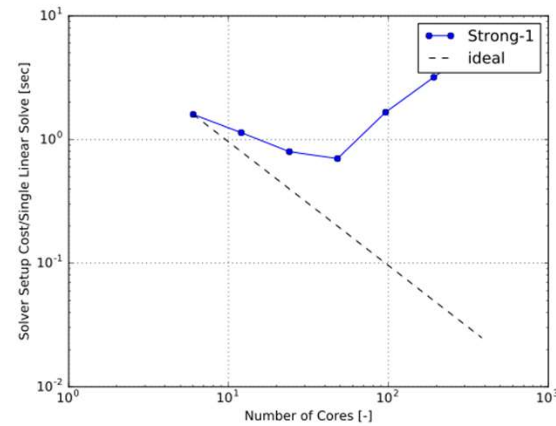
Strong Scaling

Total times

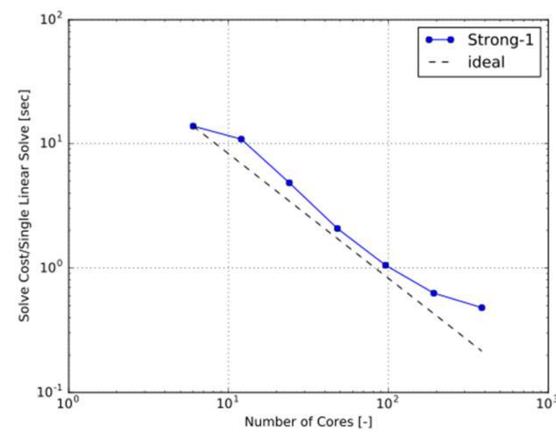




Matrix



Solver

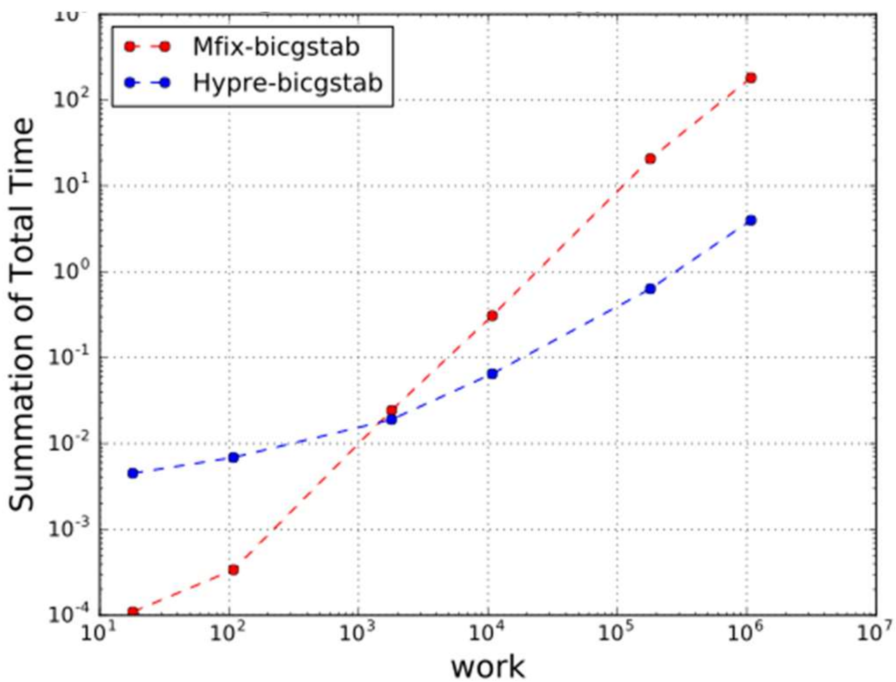


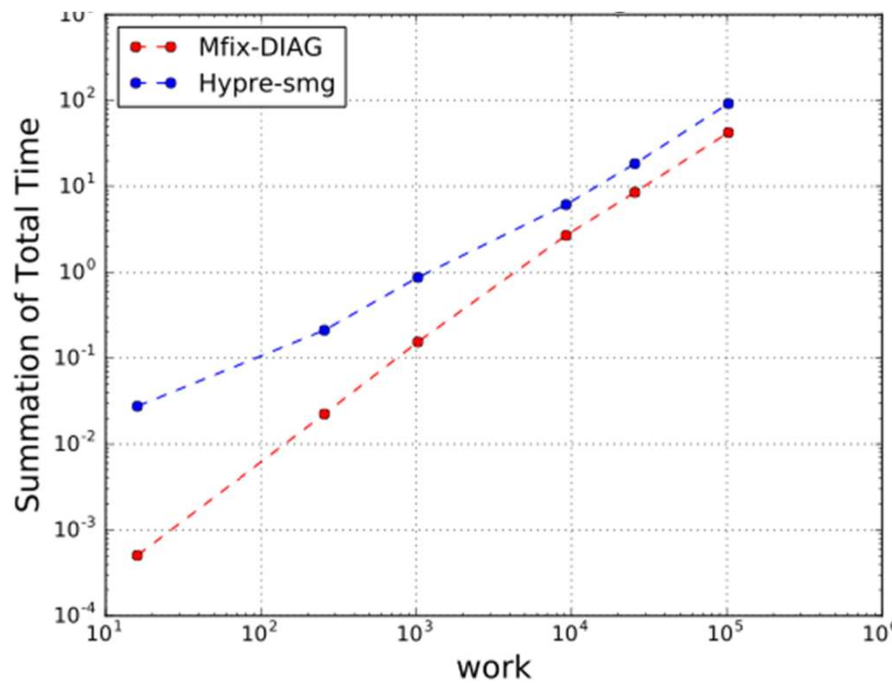
Solve

Exploratory Questions/Observation

- Setup the solver less frequently? Effect on overall convergence?
- Do all solvers have a large setup cost?
- <u>More work/Core!</u>

Comparison with MFiX





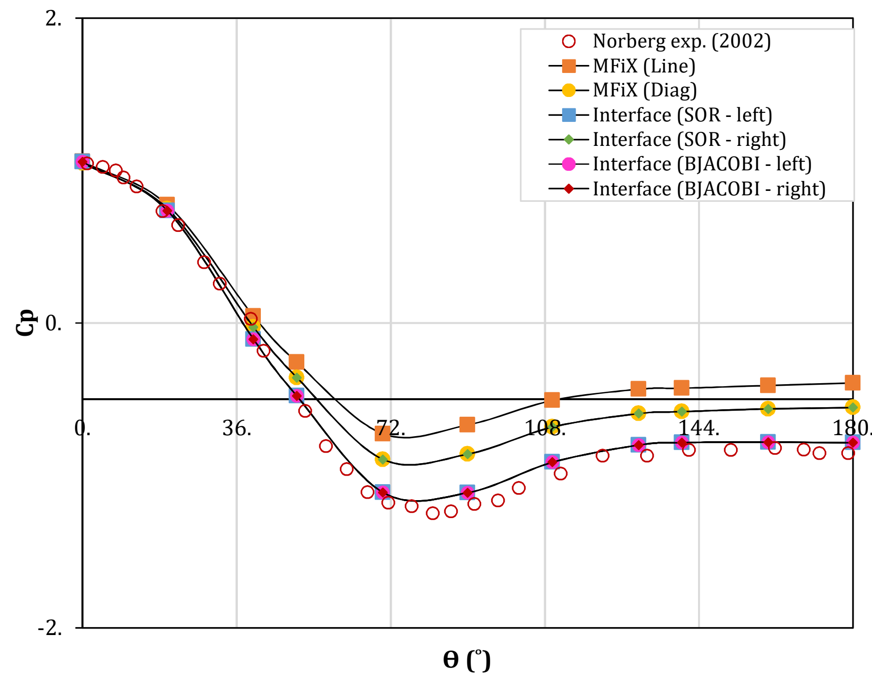
- Attempts at an apples-to-apple comparison for several test problems
- Varies problem to problem
- Work/core must be large enough for Hypre to be competitive
- Lots of knobs - only just a few have been explored up until now

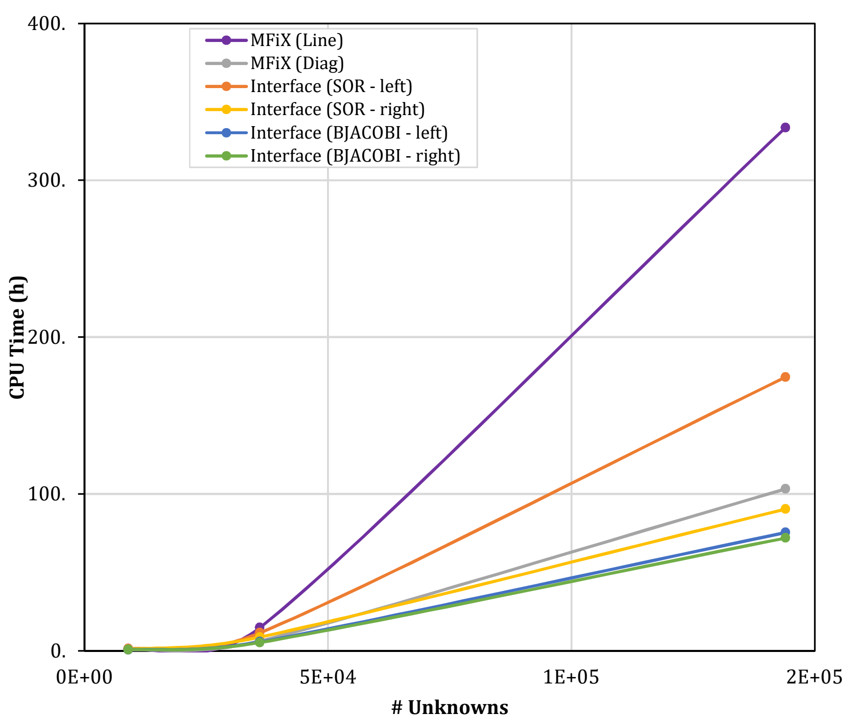
Differencing Scheme

Superbee (480x320)

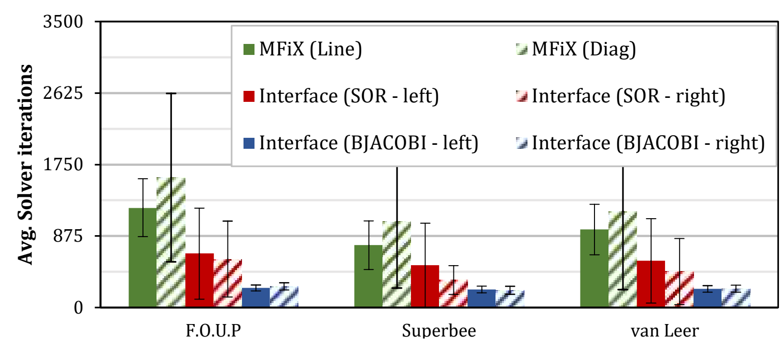
Comparison with MFiX

- Left and Right preconditioning agrees well with the data.
- Finer mesh resolutions highlight the difference between the solvers

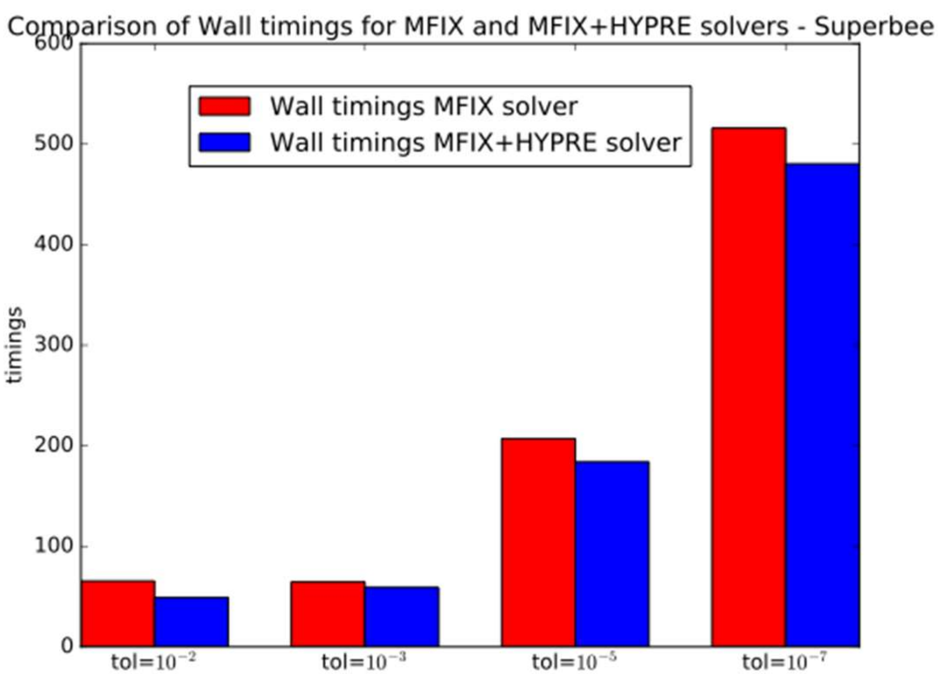


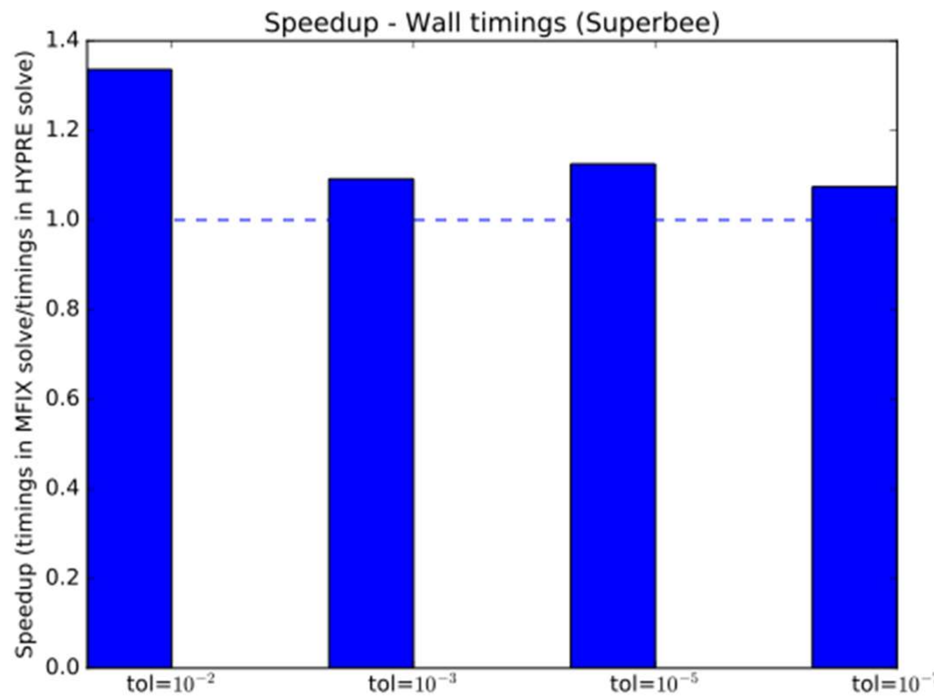


- Efficiency of the linear solver is also highlighted.



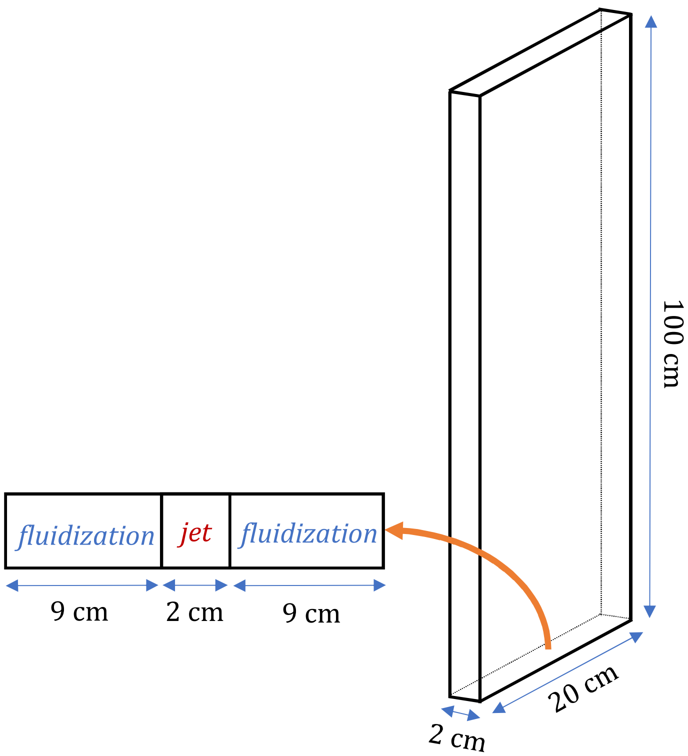
Solution Efficiency

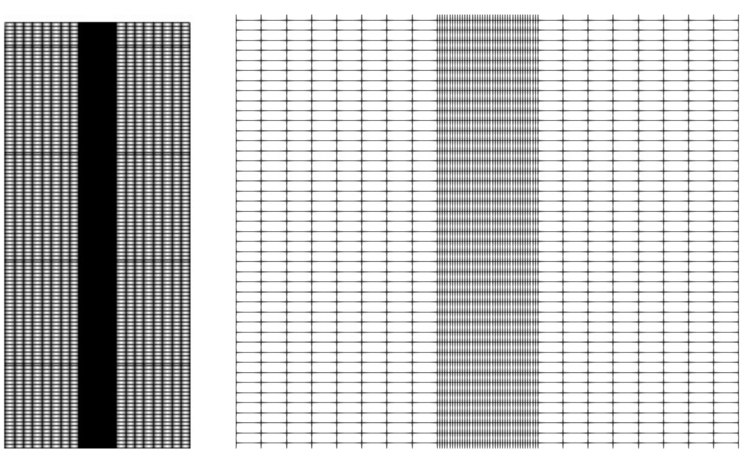




- Using flow-over-a-cylinder problem - exploring the stiffness of the linear system using various convection schemes.
- Changing intermediate parameters (wide space)
- Efficiencies are gained by tuning several adjustable solver parameters (tolerance, tolerances, multigrid parameters, etc)

3D Fluidized Bed – Polypropylene



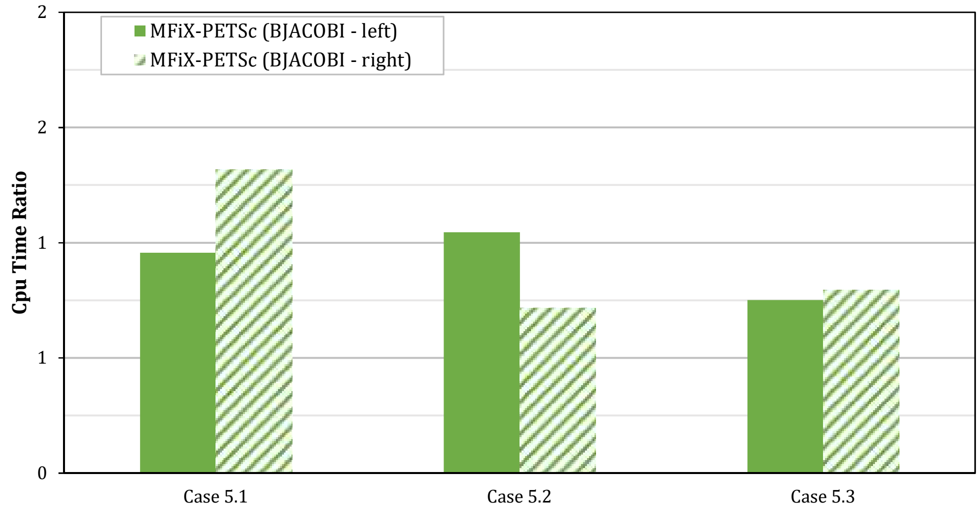


Case 4: Fluidized Bed with Polypropylene Particles (3-Dimensional)							
Case	**Dimensions + Mesh**	**Time Step**	**Tolerance**	**U_{in}**	**Solver**	**Scheme**	**P.C.**
4.1	20x100x2 cm^3 40x250x10	DT: 10^{-3} Max: 10^{-1} Min: 10^{-6}	Outer: 10^{-1} Solver: 10^{-3}	5 m/s	BCGS	van Leer	1. MFiX Line 2. PETSc BJACOBI (left) 3. PETSc BJACOBI (right)
4.2	20x100x2 cm^3 40x250x10	DT: 10^{-3} Max: 10^{-1} Min: 10^{-6}	Outer: 10^{-1} Solver: 10^{-3}	20 m/s	BCGS	van Leer	1. MFiX Line 2. PETSc BJACOBI (left) 3. PETSc BJACOBI (right)
4.3	20x100x2 cm^3 40x250x10	DT: 10^{-3} Max: 10^{-4} Min: 10^{-6}	Outer: 10^{-1} Solver: 10^{-1}	5 m/s	BCGS	van Leer	1. MFiX Line 2. PETSc BJACOBI (left) 3. PETSc BJACOBI (right)

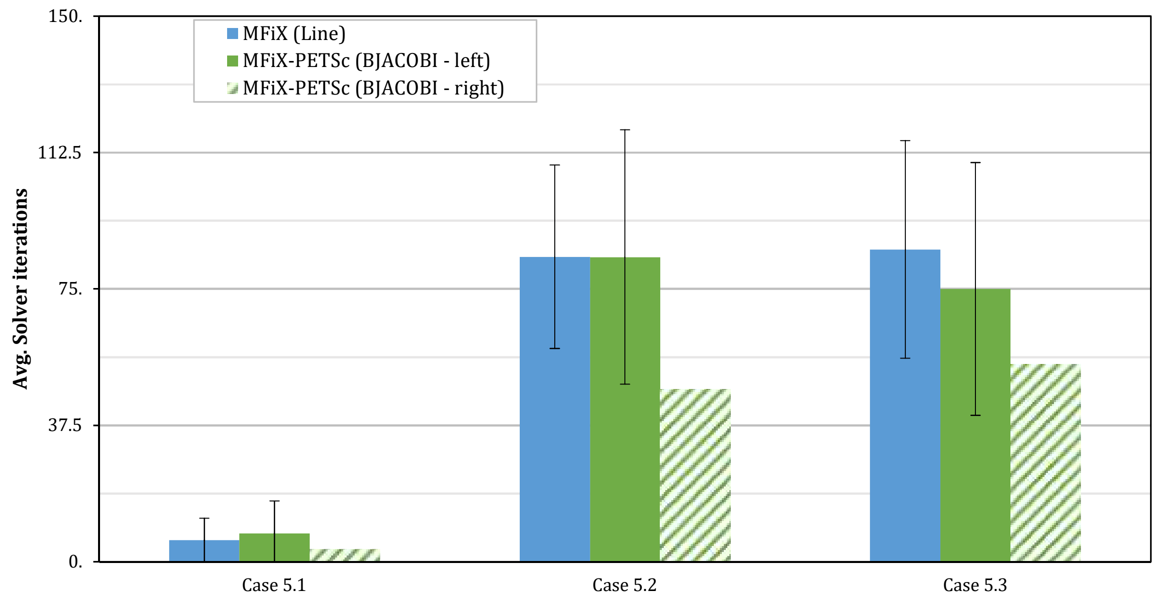
Case 5: Fluidized Bed with Polypropylene Particles (2-Dimensional)							
Case	**Dimensions + Mesh**	**Time Step**	**Tolerance**	**U_{in}**	**Solver**	**Scheme**	**P.C.**
5.1	20x100 cm^2 56x250	DT: 10^{-3} Max: 10^{-3} Min: 10^{-6}	Outer: 10^{-1} Solver: 10^{-1}	5 m/s	BCGS	van Leer	1. MFiX Line 2. PETSc BJACOBI (left) 3. PETSc BJACOBI (right)
5.2	20x100 cm^2 56x250	DT: 10^{-3} Max: 10^{-3} Min: 10^{-6}	Outer: 10^{-1} Solver: 10^{-3}	5 m/s	BCGS	van Leer	1. MFiX Line 2. PETSc BJACOBI (left) 3. PETSc BJACOBI (right)
5.3	20x100 cm^2 56x250	DT: 10^{-3} Max: 10^{-3} Min: 10^{-6}	Outer: 10^{-3} Solver: 10^{-3}	5 m/s	BCGS	van Leer	1. MFiX Line 2. PETSc BJACOBI (left) 3. PETSc BJACOBI (right)

Fluidized Bed

CPU Time Ratio PETSc/MFiX



Inner Iterations



Example: FluidBed1

Increased resolution of original input file by a factor of four.

1st attempt with native MFiX solve: diverged. :-(

2nd attempt with gmres/smg: converged. :-)

Native MFiX

```
Time =    0.0000      Dt =  0.10942E-05
  Nit     P0         P1        U0        V0        U1        V1      Max res
    1   4.0E-05   2.0E-06   1.7E-10   2.2E-03   0.        0.         V0
    2    1.       2.1E+01   6.5E-04   2.5E-03   3.        8.         P1
    3    6.       7.4E+01   6.5E-04   8.5E-03   0.6       7.         P1
    4    2.       3.4E+01   3.7E-04   3.7E-03   0.1       0.7        P1
    5    1.       2.3E+01   2.2E-04   2.7E-03   3.1E-02   0.4        P1
    6   0.9       1.9E+01   1.5E-04   2.1E-03   1.9E-02   0.2        P1
    7    3.       2.6E+01   1.5E-04   4.9E-03   1.4E-02   0.4        P1
    8   0.9       1.7E+01   1.1E-04   2.5E-03   1.1E-02   0.2        P1
    9    1.        5.       5.6E-05   2.0E-03   5.1E-03   0.1        P1
   10   0.7        8.       6.5E-05   1.4E-03   5.3E-03   0.1        P1
   11    1.        5.       5.4E-05   1.1E-03   3.1E-03   0.1        P1
   12    1.        5.       6.9E-05   1.3E-03   3.4E-03   4.9E-02    P1
   13   2.6E+01   1.4E+02   1.3E-03   2.2E-02   0.1       1.         P1
   14    2.       3.9E+01   5.5E-04   9.9E-03   2.1E-02   0.3        P1
   15    1.       3.4E+01   3.1E-04   5.3E-03   1.2E-02   0.2        P1
t=     0.0000 Dt= 0.1094E-05 NIT= 15MbErr%= 0.3207E-05: Run diverged/stalled :-(
DT < DT_MIN.  Recovery not possible!
```

Hypre

```
Time =   0.98289E-03   Dt =   0.25812E-04
  Nit     P0        P1        U0        V0        U1        V1      Max res
    1   2.2E-06   1.4E-02   1.4E-06   5.5E-05   5.7E-04   2.5E-02    V1
    2   0.5       0.3       6.3E-06   4.8E-04   2.7E-04   1.2E-02    P0
    3   0.3       0.1       8.5E-07   5.8E-05   1.4E-04   6.1E-03    P0
    4   0.2       0.1       1.2E-06   8.1E-05   7.0E-05   3.1E-03    P0
    5   0.1       3.3E-02   1.4E-06   1.1E-04   3.6E-05   1.5E-03    P0
    6   0.1       1.7E-02   1.2E-06   9.4E-05   1.9E-05   7.8E-04    P0
    7   4.1E-02   8.5E-03   9.2E-07   7.1E-05   9.8E-06   4.0E-04    P0
    8   2.5E-02   4.3E-03   6.5E-07   5.0E-05   5.1E-06   2.0E-04    P0
    9   1.5E-02   2.2E-03   4.4E-07   3.4E-05   2.7E-06   1.0E-04    P0
   10   9.4E-03   1.1E-03   2.8E-07   2.2E-05   1.4E-06   5.3E-05    P0
   11   5.7E-03   5.6E-04   1.8E-07   1.4E-05   7.7E-07   2.7E-05    P0
   12   3.5E-03   2.8E-04   1.2E-07   9.1E-06   4.2E-07   1.4E-05    P0
   13   2.1E-03   1.4E-04   7.2E-08   5.7E-06   2.3E-07   7.3E-06    P0
   14   1.3E-03   7.2E-05   4.5E-08   3.5E-06   1.2E-07   3.8E-06    P0
   15   7.8E-04   3.6E-05   2.7E-08   2.2E-06   6.9E-08   2.0E-06    P0
DT= 0.2868E-04    NIT/s=581131

t=     0.001009 Wrote SPx: 1,2,3,4,5,6,7,8,9,A,B; .OUT; .USR: 1,2,3,4,5;

.RES;

 ***********************************************************************
 Total CPU used = 59.69 s
 Total CPU IO used = 0.2428 s
 Total wall time used = 60.37 s
 ***********************************************************************
```

Summary

- Several test problems have been explored and tested for **correctness** (passed), both multicore and single core.
- **Performance** has been fairly well characterized on a series of single phase problems, highlighting the overhead costs of Hyper.
- **Comparisons** with MFiX are favorable, but may depend on the scenario and require solver parameter tuning, of which there are a few in Hypre, especially the multigrid parameters.
- Some cases better **algorithmic scaling** when compared to the use of the native MFiX solver.

Future Work

- Include more **complexity** in the problems we are exploring (spouted beds for CLC, etc.)
- **Scale-up** the problems to larger core counts
- Look for ways to **amortize** setup costs
- Other efficiency gains? **OpenMP**?
- Discussion with **MFiX team**: What problems would they like to see?