High Fidelity Computational Model for Fluidized Bed Experiments

A. Rodriguez¹, A. Schiaffino², A. Chattopadhyay², V. Kotteda³, V. Kumar⁴, W. Spotz⁵
¹UG RA, ²Graduate Student, ³Postdoc fellow
⁴Associate Professor, Mechanical Engineering
The University of Texas, El Paso

⁵Senior Staff Scientist, Sandia National Labs - NM
Outline

• Background
• Goals and Objectives
• Project tasks, milestones, and schedule
• Technical approach
• Project status – results and discussions
• Accomplishments
• Concluding remarks
Fluidized Bed Reactor

- Greater surface area of contact for fluid and solid allowing for better mixing.
- Applications in process industry and nuclear engineering.

- Governing dynamics is very complicated.
- Multiple phases, interaction between particles and fluid, varying sizes of the particles makes it very difficult to predict the behavior of the bed.

The arrival time of a space probe traveling to Saturn can be predicted more accurately than the behavior of a fluidized bed chemical reactor! (Geldart, 1986)
Supercomputers

https://www.top500.org/
The Multiphase Flow with Interphase eXchanges (MFiX) software package is
• a multiphase Computational Fluid Dynamics (CFD) software
• developed by NETL (Opensource)
• a legacy code written in Fortran
Multphase Flow with Interphase eXchange (MFiX)

• CFD code for modeling reacting multiphase systems.

• 30 years of development history with a wide range of customers.

• Successfully modeling the complexity of fluidized bed reactor flow.

• 40% of value added by the U.S chemical industry is related to particle technology- Ennis et al. Chem Eng. Progress 1994.

• Scale up of fluidized beds is a daunting task- Knowlton et al. Powder technology, 2005.
**Multphase Flow with Interphase eXchange (MFiX)**

Gas-solids are addressed by solving
- coupled continuity
- momentum conservation
- energy equations, and
- parameterizing many effects such
  - drag force, buoyancy,
  - virtual mass effect,
  - lift force,
  - Magnus force,
  - Basset force,
  - Faxen force, etc.

Provides a suite of models that treat the carrier phase (gas phase) and disperse phase (solids phase) differently.
- MFiX-TFM (Two-Fluid Model)
- MFiX-DEM (Discrete Element Model)
- MFiX-PIC (Multiphase Particle in Cell)
In nutshell, though MFiX is widely used by the fossil fuel reactor communities to model and understand the multiphase physics in a circulating fluidized, its overall utility of the multiphase models remains limited due to the computational expense of large scale simulations. The time-to-solution however can be reduced by leveraging state-of-the-art preconditions and linear solver libraries where majority of processor-level time is spent in solving large systems of linearized equations.
Challenges with MFiX

- Poor Convergence, especially in complex non-linear problems.
- Increased number of Iterations as a result of poor convergence.
- Basic preconditioners.
- Not as scalable as one would like.

\[
\frac{\partial}{\partial t} (\varepsilon_g \rho_g) + \nabla \cdot (\varepsilon_g \rho_g \vec{v}_g) = \sum_{n=1}^{N_{\text{Bi}}} R_{g, n} \\
\frac{\partial}{\partial t} (\varepsilon_g \rho_g \vec{v}_g) + \nabla \cdot (\varepsilon_g \rho_g \vec{v}_g \vec{v}_g) = \nabla \cdot \left( \vec{S}_g + \varepsilon_g \rho_g \vec{g} \right) - \sum_{n=1}^{N_{\text{Bi}}} \vec{f}_{g, n} + \vec{f}_g \\
\frac{\partial}{\partial t} (\varepsilon_{\text{sm}} \rho_{\text{sm}}) + \nabla \cdot (\varepsilon_{\text{sm}} \rho_{\text{sm}} \vec{v}_{\text{sm}}) = \sum_{n=1}^{N_{\text{Bi}}} R_{\text{sm}, n}
\]

\[
A \ x = B
\]
Technical goal

The technical goal of this project is to develop, validate and implement advanced linear solvers to replace the existing linear solvers that are used by the National Energy Technology Laboratory’s (NETL) open source software package Multiphase Flow with Interphase eXchanges (MFIX). This goal will be achieved by integrating Trilinos, a publicly available open-source linear equation solver library developed by Sandia National Laboratory. The project will demonstrate scalability of the Trilinos-MFIX interface on various high-performance computing (HPC) facilities including the ones funded by the Department of Energy (DOE).

The expected results of the project will be reduction of computational time when solving complex gas-solid flow and reaction problems in MFIX, and reduction in time and cost of adding new algorithms and physics based models into MFIX.
Objectives

• Create a framework to integrate the existing MFIX linear solver with Trilinos linear solver packages,

• Evaluate the performance of the state-of-the-art preconditions and linear solver libraries in Trilinos with MFIX, and

• Test three dimensional (3D) MFIX suites of problems on massively parallel computers with and without GPU acceleration.
Proposed Tasks and subtasks
Tasks

Task 1.0 – Project Management and Planning
Task 2.0 – Assembly of Optimum Trilinos Linear Equation Package for Integration with MFIX
   Subtask 2.1: Setup a GIT/version-control repository
   Subtask 2.2: Select the optimum Trilinos software package
   Subtask 2.3: Develop a ForTrilinos based Fortran interface for MFIX
Task 3.0 – Performance Evaluation of Preconditions and Linear Solver Libraries
   Subtask 3.1: Test and compare the linear equation solver packages in Trilinos
   Subtask 3.2: Perform a scalability analysis of Trilinos-MFIX
   Subtask 3.3: Improve performance of Trilinos-MFIX
Task 4.0 – Performance Evaluation of MFIX with the Trilinos Linear Solver on Massively Parallel Computers
   Subtask 4.1: Secure computational time on massively parallel computers
   Subtask 4.2: Compile Trilinos-MFIX on the selected massively parallel computer(s)
   Subtask 4.3: Run simulations of a fluidized bed test problems with various particle sizes and shapes
# Project milestones, budget and schedule

<table>
<thead>
<tr>
<th>Budget Year 1:</th>
<th>Title</th>
<th>Description</th>
<th>Related task or subtask</th>
<th>Expected Completion Date</th>
<th>Success Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milestone 1.1</td>
<td>GIT repository setup completed</td>
<td>Setup GIT/version-control repository for Trilinos MFIX</td>
<td>Subtask 2.1</td>
<td>Q1</td>
<td>A working repository tested by at least three researchers</td>
</tr>
<tr>
<td>Milestone 1.2</td>
<td>Best Trilinos linear solver package decided</td>
<td>Choose the best Trilinos linear solver package for MFIX</td>
<td>Subtask 2.2</td>
<td>Q2</td>
<td>Source code for the decided package uploaded to the GIT repository</td>
</tr>
<tr>
<td>Milestone 1.3</td>
<td>Fortran interface for Trilinos MFIX created</td>
<td>Develop Fortran interface for the Trilinos linear solver to communicate with MFIX</td>
<td>Subtask 2.3</td>
<td>Q3-4</td>
<td>A working version of the Fortran interface uploaded</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Budget Year 2:</th>
<th>Title</th>
<th>Description</th>
<th>Related task or subtask</th>
<th>Expected Completion Date</th>
<th>Success Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milestone 2.1</td>
<td>&gt;30% Linear solved speedup achieved</td>
<td>Test the linear solvers for its performance</td>
<td>Subtask 3.1</td>
<td>Q5</td>
<td>20% or better linear solver speedup achieved</td>
</tr>
<tr>
<td>Milestone 2.2</td>
<td>Scalability issues identified</td>
<td>Perform scalability analysis of Trilinos-MFIX</td>
<td>Subtask 3.2</td>
<td>Q6</td>
<td>Scalability testing for up to 1024 cores performed</td>
</tr>
<tr>
<td>Milestone 2.3</td>
<td>Bottlenecks to scalability identified and removed</td>
<td>Perform code profiling and identify bottlenecks</td>
<td>Subtask 3.3</td>
<td>Q7-8</td>
<td>Code profiling on Trilinos profiler completed and bottlenecks addressed for one HPC system</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Budget Year 3:</th>
<th>Title</th>
<th>Description</th>
<th>Related task or subtask</th>
<th>Expected Completion Date</th>
<th>Success Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milestone 3.1</td>
<td>Trilinos MFIX compiled on various OS/architectures</td>
<td>Compile Trilinos-MFIX on various cloud/HPC computers</td>
<td>Subtask 4.3</td>
<td>Q9</td>
<td>Trilinos MFIX compiled on 3 HPC (UTEP, DOE-Sandia, and one more)</td>
</tr>
<tr>
<td>Milestone 3.2</td>
<td>MFIX tests suites completed</td>
<td>Run simulations with various particle sizes and shapes of fluidized bed riser test problems</td>
<td>Subtask 4.3</td>
<td>Q10</td>
<td>All 2D runs and one 3D tests validated</td>
</tr>
<tr>
<td>Milestone 3.2</td>
<td>Trilinos MFIX performance analysis completed</td>
<td>Analyze Trilinos-MFIX performance for various computing architectures and fluidized-test problems</td>
<td>Subtask 4.3</td>
<td>Q11-12</td>
<td>Report submitted</td>
</tr>
</tbody>
</table>
## Project schedule

<table>
<thead>
<tr>
<th>Task Title</th>
<th>Budget Period: 1st half</th>
<th>Budget Period: 2nd half</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q1 Q2 Q3 Q4 Q5 Q6</td>
<td>Q7 Q8 Q9 Q10 Q11 Q12</td>
</tr>
<tr>
<td><strong>Program Management</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Software preparations</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subtask 2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subtask 2.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subtask 2.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Trilinos MFIX Linear Solver</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subtask 3.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subtask 3.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subtask 3.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MFIX suite tests</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subtask 4.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subtask 4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subtask 4.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Project risks & risk management plan

<table>
<thead>
<tr>
<th>Risk Category</th>
<th>Description of the Risk</th>
<th>Probability of Occurring</th>
<th>Impact</th>
<th>Overall Level of the Risk</th>
<th>Risk Mitigation Strategy</th>
</tr>
</thead>
</table>
| Internal Data Storage Array | How will the internal data storage arrays (a setpadiagonal matrix) be reconciled with native Trilinos data structures | Med | Low | Low | 1) Define a class that directly copy the data  
2) Compressed sparse row matrix storage  
3) Discussions with experts at Sandia to determine which approach would be best |
| ForTrilinos | Trilinos Fortran interfaces | High | Low | Low | 1) Consider Python as a glue language since it provides automatic wrapper generators.  
2) Leverage from collaborators’ existing PyTrilinos project  
3) Integrate the fortran interface fully with Stratimikos |
| External supercomputer access | Securing access to the external supercomputers | Low | Low | Low | Added as subtask to Secure external resources. Strategies are  
1) Work with Sandia Collaborator to secure DOE’s HPC for Trilinos MFIX performance testing  
2) Request for more allocation on TACC HPC through UT System HPC initiatives  
3) Write proposal to secure computing hours on XSEDE HPC resources  
4) PI already has access to Linux and IBM clusters through UTEP’s HPC facilities |
Project status

• Setup GIT/version-control repository
• Select optimum Trilinos software package
• Develop ForTrilinos based Fortran interface for MFIX
• Test and compare the linear solver packages in Trilinos
  • Run test cases of fluidized bed simulations in MFIX using the various Trilinos linear equations preconditioner. Compare the performance (computing time and accuracy) of the MFIX- Trilinos package with solutions that use MFIX and its existing linear solvers
• Perform scalability analysis of Trilinos-MFIX
  • Conduct scalability tests of the Trilinos-MFIX package on single node and multi-core computer clusters using distributed/shared or in hybrid environment HPC systems. Test multi-core clusters containing 4, 16, 64, 128, 512, 1024, 8192(?) cores.
• Address Trilinos-MFIX performance bottlenecks via profiling and debugging tools in Trilinos
• Run fluidized bed test problems (various particle sizes and shapes, 2D/3D, Small/Large Scale, etc.)
Technical approach

One of the main challenges for any software development is keeping the computer code up-to-date with the advancement in applied mathematics, software and hardware development in computational science and engineering. Realizing the challenge, the Computer Science Research Institute (CSRI) group at Sandia National Laboratories (Sandia) has developed and continues to develop scalable solver algorithms and software through next-gen (exa-scale, peta-scale, exteme-scale, etc.) computing investment. The project is called Trilinos project.

Funded by various DOE entities mainly NNSA - Advanced Simulation and Computing (ASC)/DOE Office of Science (SciDAC), Advanced Scientific Computing Research (ASCR)

Note: Slides in this topic mostly borrowed from M.Heroux & other trilinos members
The Trilinos Project is an effort to develop and implement robust algorithms and enabling technologies using modern object-oriented software design, while still leveraging the value of established libraries such as PETSc, Metis/ParMetis, SuperLU, Aztec, the BLAS and LAPACK. It emphasizes abstract interfaces for maximum flexibility of component interchanging, and provides a full-featured set of concrete classes that implement all abstract interfaces. Research efforts in advanced solution algorithms and parallel solver libraries have historically had a large impact on engineering and scientific computing. Algorithmic advances increase the range of tractable problems and reduce the cost of solving existing problems. Well-designed solver libraries provide a mechanism for leveraging solver development across a broad set of applications and minimize the cost of solver integration. Emphasis is required in both new algorithms and new software (Heroux et.al., http://trilinos.sandia.gov/).
What is Trilinos?

- Object-oriented software framework for...
- Solving big complex science & engineering problems
- More like LEGO™ bricks than Matlab™

**Trilinos** provides the state-of-the-art preconditions and linear solver libraries
- demonstrate **scalability** on **current HPC systems**
- illustrate plans for **continued maintenance**
- include **support for new hardware technologies**
Target Platforms

Desktop: Development and more...

Capability machines:
  Redstorm (XT3), Clusters
  Roadrunner (Cell-based).

**Multicore nodes.**

Parallel software environments:
  MPI
  UPC, CAF, threads, vectors,...
  Combinations of the above.

User “skins”:  
  C++/C, Python
  Fortran.

Web, CCA.
Evolving Trilinos Solution

- Beyond a "solvers" framework
- Natural expansion of capabilities to satisfy application and research needs

**physics**

\[ L(u) = f \]
Math. model

\[ L_h(u_h) = f_h \]
Numerical model

\[ u_h = L_h^{-1} f_h \]
Algorithms

**computation**

\[ \mathbf{Numerical \ math} \]
Convert to models that can be solved on digital computers

\[ \mathbf{Algorithms} \]
Find faster and more efficient ways to solve numerical models

\[ \mathbf{discretizations} \]
Time domain
Space domain

\[ \mathbf{methods} \]
Automatic diff.
Domain dec.
Mortar methods

\[ \mathbf{solvers} \]
Linear
Nonlinear
Eigenvalues
Optimization

\[ \mathbf{Trilinos \ core} \]
Petra
Utilities
Interfaces
Load Balancing

- Discretization methods, AD, Mortar methods, …
Trilinos Strategic Goals

**Scalable Computations:** As problem size and processor counts increase, the cost of the computation will remain nearly fixed.

**Hardened Computations:** Never fail unless problem essentially intractable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.

**Full Vertical Coverage:** Provide leading edge enabling technologies through the entire technical application software stack: from problem construction, solution, analysis to optimization.

**Grand Universal Interoperability:** All Trilinos packages will be interoperable, so that any combination of packages that makes sense algorithmically will be possible within Trilinos and with compatible external software.

**Universal Accessibility:** All Trilinos capabilities will be available to users of major computing environments: C++, Fortran, Python and the Web, and from the desktop to the latest scalable systems.

**Universal Capabilities RAS:** Trilinos will be:

- The leading edge hardened, efficient, scalable solution for each of these applications (**Reliability**).
- Integrated into every major application at Sandia (**Availability**).
- Easy to maintain and upgrade within the application environment (**Serviceability**).
C++ Objected Oriented Framework

Flexibility in terms of generic programming with Tpetra

Parallel/Distributed Templated classes for matrix generations

State-of-The-Art Linear and Non-Linear Solvers

Run time load balancing between CPU and GPU via Kokkos

Native support for exascale (>2 billion unknowns) computing

Hardware Exploitation for large scale number crunching

Parallel Data Decomposition data structures Map
Objective

Cross Language Integration of CFD code with faster solvers and algebraic preconditioners

Advantages

- Exploit legacy codes' expertise in setting up large scale problems
- Use Trilinos as a faster and modern solver platform integrated with legacy codes.

Challenges

- FORTRAN has no objected oriented framework.
- Trilinos has been developed in C++ (objected oriented) framework.
- No semantic support for C++ in FORTRAN.
Flow chart

LEQ Method Description

1. SOR - Point Successive Over Relaxation
2. BiCGSTAB - Bi-Conjugate Gradient STABilized method
3. GMRES - Generalized Minimal RESidual algorithm
4. BICGSTAB + GMRES
5. CG - Conjugate Gradient
6. BiCGSTAB/GMRES/CG/Direct/... - Trilinos

Initialization of computations
Decompose the domain based on NODESI, NODESJ and NODESK
Apply BC and solve the system of equations for fluid flow variables
Output

Yes
Stop

No
Finished time steps

Yes

LEQ < 6

MFiX

Interface

Trilinos

Flow chart
Cross Language Integration Scheme

M-FIX

Fortran Wrapper

A b D

C-Wrapper
Extern "C"

Trilinos

Epetra A(Map)
Epetra b(Map)
Map=D

CPP Wrapper

A b D
A language independent interface to integrate legacy codes

Transfer A & b
Define solver attributes

LEQ = 6

Transfer x

MFiX wrapper
Interpret matrix structure, Implement CRS scheme

Fortran wrapper
Semantic for memory references

C wrapper
Communicator, Interpret Polymorphism representations

Cpp wrapper
Create Epetra_MAP, Fill A & b

Solution of Ax=b

Trilinos
Loop number of time steps
Loop number of nonlinear iterations
If LEQ=6
Solve linear equations governing the fluid and solid/bubbles using Trilinos solver
Else
Solve using MFiX solvers
End
End
End

If LEQ=6
Solve linear equations governing the fluid and solid/bubbles using Trilinos solver
Else
Solve using MFiX solvers
End
End
End
nstart = istart;  nend = iend

ie = 0
  do k = kstart,kend
    do i = nstart,nend
      do j = jstart, jend
        IJK = funijk(i,j,k)
        IJK_GL = funijk_gl(i,j,k)
        ie = ie + 1
        Anew(ie,1) = A_M(IJK,-3)
        Anew(ie,2) = A_M(IJK,-1)
        Anew(ie,3) = A_M(IJK,-2)
        Anew(ie,4) = A_M(IJK,0)
        Anew(ie,5) = A_M(IJK,2)
        Anew(ie,6) = A_M(IJK,1)
        Anew(ie,7) = A_M(IJK,3)
        Bn(ie) = B_M(IJK)
        locgl(ie) = IJK_GL
        pos(ie,1) = KM_OF(IJK) - IJK
        pos(ie,2) = IM_OF(IJK) - IJK
        pos(ie,3) = JM_OF(IJK) - IJK
        pos(ie,4) = JP_OF(IJK) - IJK
        pos(ie,5) = IP_OF(IJK) - IJK
        pos(ie,6) = KP_OF(IJK) - IJK
      enddo
    enddo
  enddo
Tpetra_Map map(numGlobalElements, numMyElements, indexBase, comm)

Tpetra_crsMatrix A(map,7);

.Tpetra_multivector x (map,1);
.Tpetra_multivector b (map,1);

for (LO i = 0; i < static_cast<LO> (numMyElements); ++i) {
   Values = Anew[][];
   Indices = pos[][];
   A->insertGlobalValues (gblRow, NumEntries, Values, Indices);
}
A->fillComplete (map,map);

for (LO i= 0; i < static_cast<LO> (numMyElements); ++i) {
   const GO gblRow = map->getGlobalElement (i);
   b->sumIntoGlobalValue(gblRow, 0, Bn[i]);
}

.Tpetra_LinearProblem problem(&A, &x, &b);

Problem->setRightPrec (plistp);

belos_bicgstab_manager_type solver(Problem, plists));

 solver->solve();
Flow chart of Cpp wrapper
for preconditioned iterative solver in Belos
Abstract Numerical Algorithms

An abstract numerical algorithm (ANA) is a numerical algorithm that can be expressed solely in terms of vectors, vector spaces, and linear operators.

**Example Linear ANA (LANA) : Linear Conjugate Gradients**

Given:
- $A \in \mathcal{X} \rightarrow \mathcal{X}$: s.p.d. linear operator
- $b \in \mathcal{X}$: right hand side vector

Find vector $x \in \mathcal{X}$ that solves $Ax = b$

**Linear Conjugate Gradient Algorithm**

Compute $r^{(0)} = b - Ax^{(0)}$ for the initial guess $x^{(0)}$.

for $i = 1, 2, \ldots$

1. $\rho_{i-1} = \langle r^{(i-1)}, r^{(i-1)} \rangle$
2. $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$ ($\beta_0 = 0$)
3. $p^{(i)} = r^{(i-1)} + \beta_{i-1}p^{(i-1)}$ ($p^{(1)} = r^{(1)}$)
4. $q^{(i)} = Ap^{(i)}$
5. $\gamma_i = \langle p^{(i)}, q^{(i)} \rangle$
6. $\alpha_i = \rho_{i-1}/\gamma_i$
7. $x^{(i)} = x^{(i-1)} + \alpha_ip^{(i)}$
8. $r^{(i)} = r^{(i-1)} - \alpha_iq^{(i)}$

check convergence; continue if necessary

- **Types of operations**
  - linear operator applications
  - vector-vector operations
  - scalar operations
  - scalar product $\langle x, y \rangle$ defined by vector space

- **Types of objects**
  - Linear Operators
    - $A$
  - Vectors
    - $r, x, p, q$
  - Scalars
    - $\rho, \beta, \gamma, \alpha$
  - Vector spaces?
    - $\mathcal{X}$

- ANAs can be very mathematically sophisticated!
- ANAs can be extremely reusable!
Gas-solid flow in a 2D bubbling fluidized bed

Cartesian vessel: 10cm length and 100cm height

Sand particle diameter = 0.04cm
Sand particle density = 2.0 g/cm³
Restitution co-efficient = 0.80
Angle of internal friction = 30
The minimum void fraction = 0.42
Fluid viscosity = 0.00018 g/cm s
Fluid density = 0.0012 g/cm³

Boundary conditions
Inlet: constant mass inflow
124.6 cm/s for 4.3<x<5.7; 25.9cm/s for 0<x<4.3, 5.7<x<10
Sidewalls: slip condition
Outlet : pressure outflow condition (p = 0)

Mesh
Structured: IMAX = 10, JMAX= 100
Flow (void fraction) in a fluidized bed

MFiX

MFiX-Trilinos
Gas-solid flow in a 3D bubbling fluidized bed

Cartesian vessel: 10cm length, 10cm width and 100cm height.

- Sand particle diameter = 0.04cm
- Sand particle density = 2.0 g/cm³
- Restitution co-efficient = 0.80
- Angle of internal friction = 30
- The minimum void fraction = 0.42
- Fluid (gas) viscosity = 0.00018 g/cm s
- Fluid density = 0.0012 g/cm³

**Boundary conditions**

- Inlet: constant mass inflow (124.6 cm/s for 4.3<x<5.7, 4.3<z<5.7)
- Sidewalls: slip condition
- Outlet: pressure outflow condition (p = 0)

**Mesh**

- Structured: IMAX = 10, JMAX= 100, KMAX = 10
Flow (void fraction) in a fluidized bed
Gas-solid flow in a circulating fluidized bed

Cylindrical vessel: 10cm diameter 100cm height.

Sand particle diameter = 0.04 cm
Sand particle density = 2.0 g/cm^3
Restitution co-efficient = 0.80
Angle of internal friction = 0
The minimum void fraction = 0.42
Fluid (gas) viscosity = 0.00018 g/cm s
Fluid density = 0.0012 g/cm^3

Boundary conditions
Inlet: constant mass inflow (124.6 cm/s for 0<r<0.7)
Sidewalls: slip condition
Outlet: pressure outflow condition (p = 0)

Mesh
Structured: IMAX = 10, JMAX = 100, KMAX = 10
Flow (void fraction) in a fluidized bed

MFiX

MFiX-Trilinos
Various computer architectures used for the performance analysis study

<table>
<thead>
<tr>
<th></th>
<th>Stampede (AS)</th>
<th>Comet (AC)</th>
<th>Bridges (AB)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model (Intel Xeon)</strong></td>
<td>E5-2680 2.7GHz</td>
<td>E5-2695 2.5GHz</td>
<td>E5-2695 2.30 GHz</td>
</tr>
<tr>
<td><strong>Cores per socket</strong></td>
<td>8</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td><strong>Sockets</strong></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>L1 cache (KB)</strong></td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td><strong>L2 cache (KB)</strong></td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td><strong>L3 cache (KB)</strong></td>
<td>20480</td>
<td>30720</td>
<td>35840</td>
</tr>
<tr>
<td><strong>RAM (GB)</strong></td>
<td>32</td>
<td>128</td>
<td>130</td>
</tr>
</tbody>
</table>
Performance of MFIX-Trilinos

![Graph showing performance of MFIX-Trilinos](image)

- **AS, $n_{dof}=10M$**
- **AC, $n_{dof}=10M$**
- **AC, $n_{dof}=200M$**
- **AB, $n_{dof}=10M$**
- **AB, $n_{dof}=200M$**

The graph plots the ratio $t_M/t_{MT}$ against $np$, where $t_M$ is the time taken for the MFIX-Trilinos code, and $t_{MT}$ is a reference time. The markers represent different configurations of the code with varying degrees of freedom ($n_{dof}$).
Speedup study

AS

AB

AC
Remarks:
• Trilinos linear solver was integrated with MFiX
• Scalability and speed-up tests for 2D & 3D bubbling flow problems were performed for upto 1024 processors
• >50% speed is observed but further investigation is required to examine any biases of the solver parameters

Scholarly dissemination
• V. Kotteda, A. Chattopadhyay*, V. Kumar, W. Spotz, “Next generation exascale capable multiphase solver with Trilinos”, ASME Journal of Fluid Engineering, under review
• V. Kotteda, A. Chattopadhyay*, V. Kumar, W. Spotz, “Next-generation multiphase flow solver for fluidized bed applications”, ASME FEDSM2017-69555
• A. Chattopadhyay*, V. Kumar, V. Kotteda, W. Spotz, “Next generation exascale capable multiphase solver with Trilinos”, ASME IMECE2016-67962 (2016)
• V. Kotteda, A. Chattopadhyay*, V. Kumar, W. Spotz, A Framework to Integrate MFiX with Trilinos for High Fidelity Fluidized Bed Computations, 2016 IEEE High Performance Extreme Computing (HPEC) (2016)
Q & A?

We would like to acknowledge National Energy Technology Laboratories (NETL), Sandia National Laboratories (Sandia), Texas Advanced Computing Center (TACC), eXtreme Science and Engineering Discovery Environment (XSEDE), Research cloud team at UT El Paso, Mechanical Engineering and Computational Science Programs at UT El Paso. This material is based upon work supported by the Department of Energy under Award Number DE-FE0026220.