HPC in Engineering

Feng Chen
HPC User Services
LSU HPC LONI
sys-help@loni.org

Louisiana State University
Baton Rouge
November 4, 2015

Outline

- **Things to be covered in the training**
  - Why Do Engineers Use HPC?
  - Types of Problems Solved
  - How We Solve PDEs encountered in engineering.
    - Commonly used PDE solution methods
    - Why HPC is necessary
  - Review of Selected HPC engineering codes
    - OpenFOAM (Available on HPC & LONI clusters)
    - Ansys Fluent Available on HPC clusters
    - DualSPhysics
    - LIGGGHTS
      - Coupled CFDEM (LIGGGHTS + OpenFOAM)

Why Do Engineers Use HPC?

- **Help determine viability before building expensive prototypes**
- **Avoid destructive testing on expensive materials or real models**
- **Use methods such as the adjoint method to optimize geometries for optimal lift, drag, or some other metric**
- **Test events for which testing is prohibitively expensive and/or dangerous**
- **Perform analysis that are very difficult or even impossible to obtain through experiments.**

- **Many tools of this sort run on desktops, but take days and weeks to run, making the results useless in real time.**

Examples of Engineering Problems

- **Solution of PDE for Continuum**
  - Navier-Stokes equation
    - Incompressible/Compressible flow such as airfoil, wind tunnel, etc.
  - Stress analysis
    - Simulation of structural deformation,, etc.
- **Solution of Discrete Medium**
  - Large deformation granular media
  - Model human behavior in terms of traffic and evacuation responses (e.g. under an emergency evacuation)
- **Solution of Coupled Continuum - Discrete Medium**
  - Model fluid-solid two phase flow
    - fluidized bed, concentrated suspensions, etc.

$$\vec{F} = m\vec{a}$$

Common Scenarios for Solving HPC Engineering Problems

- **Use existing commercial package**
  - e.g. ANSYS, ABAQUS
- **Use existing open source package**
  - e.g. Deal II, OpenFOAM, SU2
- **Write your own code based on existing packages**
  - Some Advantage of using open source packages
    - Easy to add/remove functions
    - Flexible for writing coupled solver/solutions

Commonly Used Parallel Scheme

- **MPI**
  - Make sure use correct version of MPI (compile time/run time)
  - Check the number of processes on each node
- **OpenMP**
  - Check the number of threads used (e.g. OMP_NUM_THREADS)
- **Hybrid MPI+OpenMP**
  - Check number of processes/number of threads
- **Using Accelerators**
  - GPU
  - Xeon Phi Co-Processor
  - FPGA, etc

Theoretical Methods to Solve PDEs

- **Mesh methods - problem domain originally defined on meshes of data points.**
  - Finite Difference Method (FDM)
  - Finite Volume Method (FVM)
  - Finite Element Method (FEM)
- **Meshless methods - do not require a mesh to connect data points of the simulation domain.**
  - Smoothed Particle Hydrodynamics (SPH)
  - Discrete Element Method (DEM)
- **Hybrid**
  - Lattice Boltzmann method (LBM)

Selected HPC Engineering packages

- **Finite Element packages**
  - ANSYS/Mechanical
  - ABAQUS
  - Deal II
- **Finite Volume packages**
  - ANSYS/Fluent
  - OpenFOAM
  - SU2
- **Meshless methods packages**
  - SPH (Smoothed Particle Hydrodynamics)
    - DualSPhysics
  - Discrete Element Method
    - LIGGGHTS, ESyS-Particle, YADE
- **Coupled Methods**
  - Euler/Langrangian

Packages To Be Introduced Today

- OpenFOAM
- Ansys/Fluent
- DualSPhysics
- LIGGGHTS
  - CFDEM (LIGGGHTS + OpenFOAM)

Introduction to OpenFOAM

- **Open Field of Operation And Manipulation (FOAM)**
  - Free, open source CFD software package
  - C++ programming language
  - A set of libraries for continuum mechanics
- **Based on Finite Volume Method (FVM)**
- **Ideal for research oriented CFD problems:**
  - Open architecture—will be detailed later
  - Low(Zero)-cost CFD
  - Problem-independent numerics and discretization
  - Efficient environment for complex physics problems

OpenFOAM

The Open Source CFD Toolbox

OpenFOAM features overview

- **Physical Modeling Capability:**
  - Basic: Laplace, potential flow, passive scalar/vector/tensor transport
  - Incompressible and compressible flow: segregated pressure-based algorithms
  - Heat transfer: buoyancy-driven flows, conjugate heat transfer
  - Multiphase: Euler-Euler, VOF free surface capturing and surface tracking
  - Pre-mixed and Diesel combustion, spray and in-cylinder flows
  - Stress analysis, fluid-structure interaction, electromagnetics, MHD, etc.

OpenFOAM features overview

- **Straightforward representation of partial differential equations (PDEs):**

$$\frac{\partial \rho U}{\partial t} + \nabla \cdot \rho U U - \nabla \cdot \mu \nabla U = -\nabla p$$

```
solve
    (
        fvm::ddt(rho, U)
      + fvm::div(phi, U)
      - fvm::laplacian(mu, U)
        ==
      - fvc::grad(p)
    );
```

Introduction to OpenFOAM

- **History of OpenFOAM:**
  - Original development started in the late 1980s at Imperial College, London (FORTRAN)
  - Later changed to C++
  - OpenFOAM 1.0 released on 10/12/2004
  - Major releases: 1.4, 1.5, 1.6, 1.7.x, 2.0.x, 2.1.x, 2.2.x, **2.3.x,** ***2.4.x***
  - Wikki Ltd. Extend-Project: 1.4-dev, 1.5-dev, ***1.6-ext***

Introduction to OpenFOAM

- **Theoretical background**
  - **Finite Volume Method (FVM)**
  - Unstructed grid
  - Pressure correction methods (SIMPLE, PISO, and their combination PIMPLE), for more information about FVM, see:

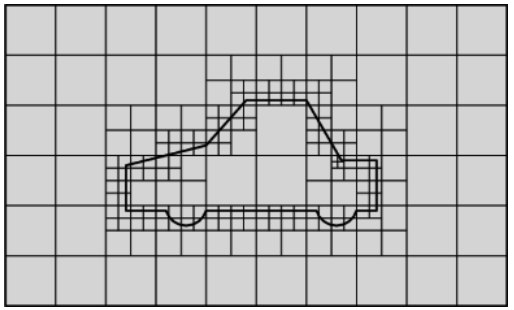
    - Partankar, S. V. (1980) *Numerical heat transfer and fluid flow*, McGraw-Hill.
    - H. Versteeg and W. Malalasekra, (2007) *An Introduction to Computational Fluid Dynamics: The Finite Volume Method Approach*
    - Ferziger, Joel H., Peric, Milovan, (2002) *Computational Methods for Fluid Dynamics*

OpenFOAM toolbox overview

- **Standard Solvers**
  - "Basic" CFD codes: e.g. laplacianFoam
  - Incompressible flow: e.g. icoFoam, simpleFoam
  - Compressible flow: e.g. rhoSimpleFoam, sonicFoam
  - Multiphase flow: e.g. interFoam
  - Direct numerical simulation (DNS) and large eddy simulation (LES)
  - Combustion
  - Particle-tracking flows (PIC)

Mesh Generation

- **blockMesh**
  - For simple geometries, there is blockMesh, a multi-block mesh generator that generates meshes of hexahedra from a text configuration file.
  - Look at the OpenFOAM distribution files which contains numerous example configuration files for blockMesh to generate meshes for flows around simple geometries, e.g. a cylinder, a wedge, etc.
- **snappyHexMesh**
  - For complex geometries, meshes to surfaces from CAD
  - Can run in parallel
  - Automatic load balancing
- **Other mesh generation tools**
  - extrudeMesh
  - polyDualMesh



Mesh Conversion

- **From: http://www.openfoam.org/features/mesh-conversion.php**

Part of the mesh converters	
ansysToFoam	Converts an *ANSYS* input mesh file, exported from *I-DEAS*, to OPENFOAM® format
cfx4ToFoam	Converts a *CFX* 4 mesh to OPENFOAM® format
datToFoam	Reads in a datToFoam mesh file and outputs a points file. Used in conjunction with blockMesh
fluent3DMeshToFoam	Converts a *Fluent* mesh to OPENFOAM® format
fluentMeshToFoam	Converts a *Fluent* mesh to OPENFOAM® format including multiple region and region boundary handling
foamMeshToFluent	Writes out the OPENFOAM® mesh in *Fluent* mesh format
foamToStarMesh	Reads an OPENFOAM® mesh and writes a *PROSTAR* (v4) bnd/cel/vrt format
foamToSurface	Reads an OPENFOAM® mesh and writes the boundaries in a surface format
gambitToFoam	Converts a *GAMBIT* mesh to OPENFOAM® format
gmshToFoam	Reads .msh file as written by Gmsh
See http://www.openfoam.org/features/mesh-conversion.php for complete list	

Running OpenFOAM on HPC/LONI clusters

- **If you're going to run it on the HPC/LONI cluster we recommend to use one of the versions already installed on our system.**
  - SuperMike2: 2.2.0
  - SuperMIC and QB2: 2.3.0
- **The package is installed as a module, so you have to load it using `module` or `softenv` before using it.**
- **However, being a free and open-source code, OpenFOAM can be modified and re-compiled by the user, if you want to run customized solver or need a particular version (e.g. OpenFOAM Ext version), please compile by yourself in home/project directory.**

Changes to your ~/.soft file

- **Add the following keys to ~/.soft and then `resoft`**
  - On SuperMike2:

```
+Intel-13.0.0
+openmpi-1.6.3-Intel-13.0.0
+OpenFOAM-2.2.1-Intel-13.0-openmpi-1.6.3
```

  - On Eric:

```
+gcc-4.7.0
+openmpi-1.6.3-gcc-4.7.0
+OpenFOAM-2.2.2-gcc-4.7.0-openmpi-1.6.3
```

  - On SuperMIC or QB2:

```
module load openfoam/2.3.0/INTEL-140-MVAPICH2-2.0
```

- **Start an interactive session:**

```
qsub -I -l nodes=1:ppn=16,walltime=02:00:00 -A your_allocation_name
```

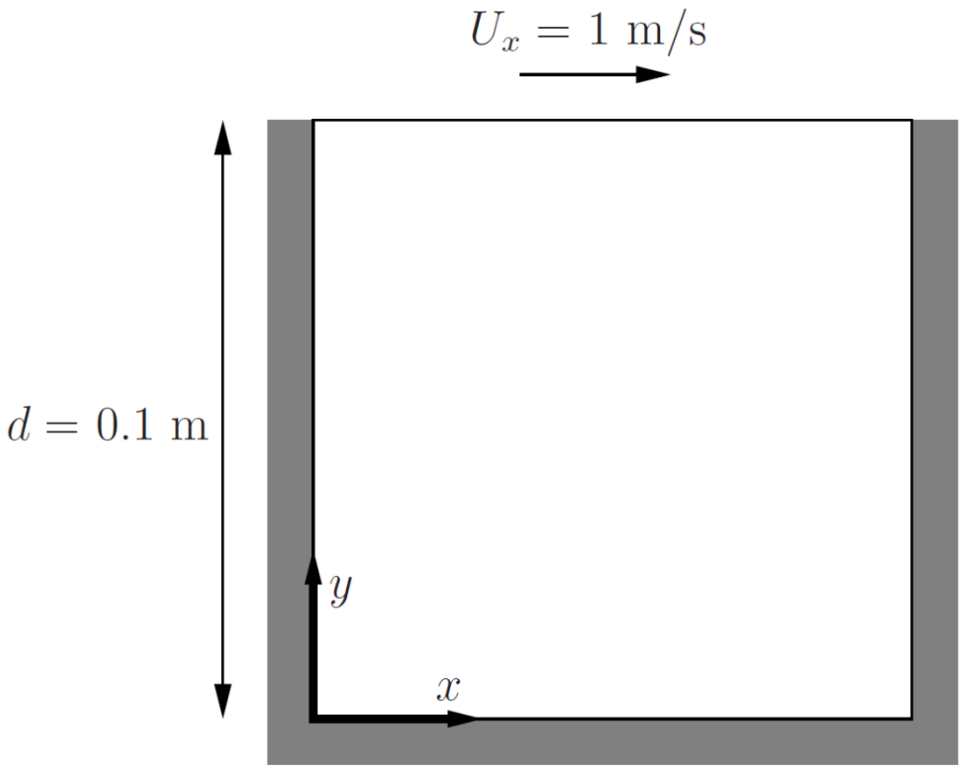
Run First OpenFOAM case

- **Steps of running first OpenFoam case on SuperMike2:**

```
$ cd /work/$USER/
$ wget https://tigerbytes2.lsu.edu/users/hpctraining/web/2015-Fall/hpcengr.tar.gz
$ tar zxf hpcengr.tar.gz
$ cd /work/$USER/hpcengr/intro_of/cavity
$ blockMesh #generate mesh information
$ icoFoam #running the PISO solver
$ foamToVTK #convert to VTK format, optional
$ paraFoam #post-processing, not suggested for large scale problems
```

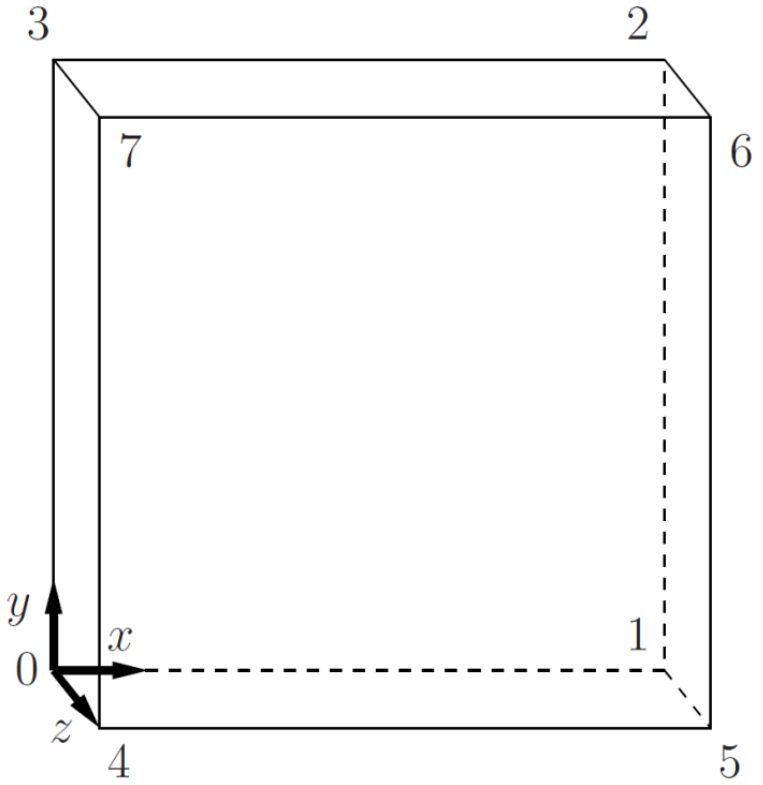
Run First OpenFOAM case

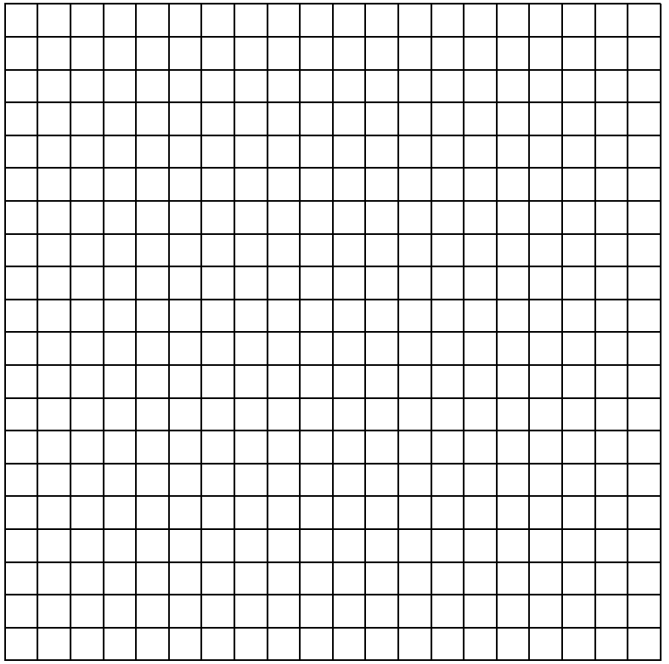
- **Lid-driven cavity flow using icoFoam**



Lid-driven cavity flow

- **The cavity domain consists of a square of side length *d=0.1m* in the *x-y* plane. A uniform mesh of 20x20 cells will be used initially.**

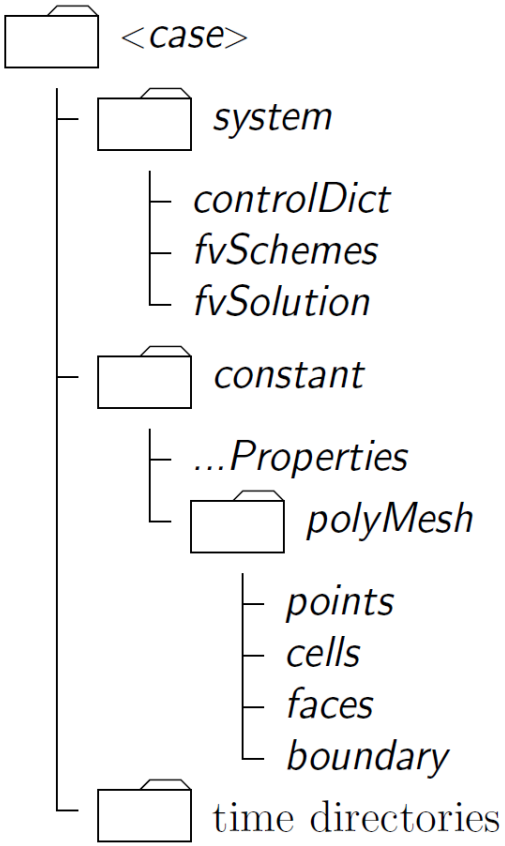




Inside case configuration

- **File structure of OpenFOAM cases**

  $ ls -R $FOAM_RUN/tutorials/incompressible/icoFoam/cavity



Inside case configuration

- **The minimum set of files required to run an OpenFOAM case**
  - constant directory:
    - description of the case mesh (geometry): e.g. ***polyMesh***
    - physical properties files: e.g. ***transportProperties***
  - system directory: solution procedure settings
    - ***controlDict***
    - ***fvSchemes***
    - ***fvSolution***
  - "time" directories: ***U, p***
    - initial conditions (I.C.)
    - boundary conditions (B.C.)
    - Future result files (typically determined by controlDict)

Inside case configuration

- **constant directory:**
  - polyMesh
    - ***blockMeshDict: mesh description, will be detailed later***
    - boundary: list of patches with BCs definition
    - faces: list of mesh faces (list of points)
    - neighbour: list of neighboring cell labels
    - owner: list of owning cell labels
    - points: list of mesh points with their coordinates
  - transportProperties

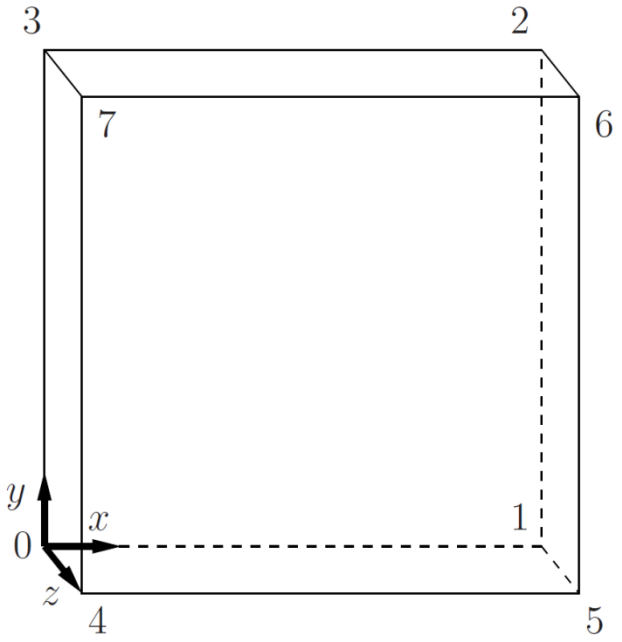
Edit blockMeshDict file (0)

- **OpenFOAM file header:**

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                      |                                            |
| \\      /  F ield              | OpenFOAM: The Open Source CFD Toolbox      |
|  \\    /   O peration          | Version:  2.2.1                            |
|   \\  /    A nd                | Web:      www.OpenFOAM.org                 |
|    \\/     M anipulation       |                                            |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
```

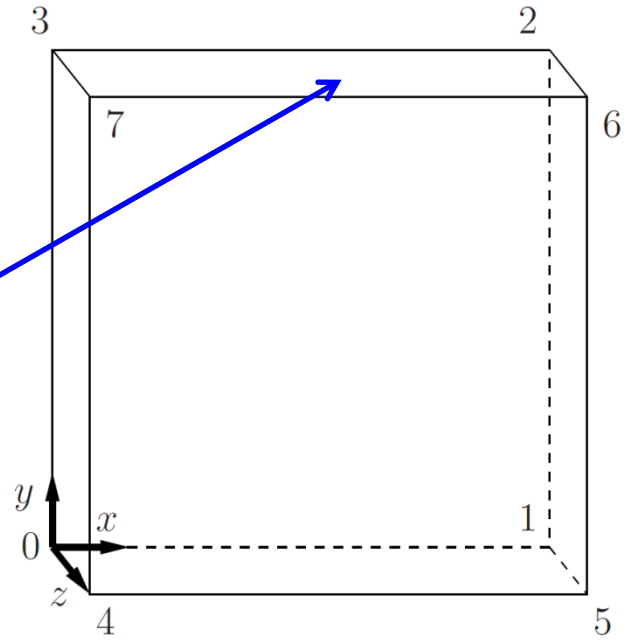
Edit blockMeshDict file (1)

```
17  convertToMeters 0.1;
18
19  vertices
20  (
21      (0 0 0)    //0
22      (1 0 0)    //1
23      (1 1 0)    //2
24      (0 1 0)    //3
25      (0 0 0.1)  //4
26      (1 0 0.1)  //5
27      (1 1 0.1)  //6
28      (0 1 0.1)  //7
29  );
30
31  blocks
32  (
33      hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
34  );
```



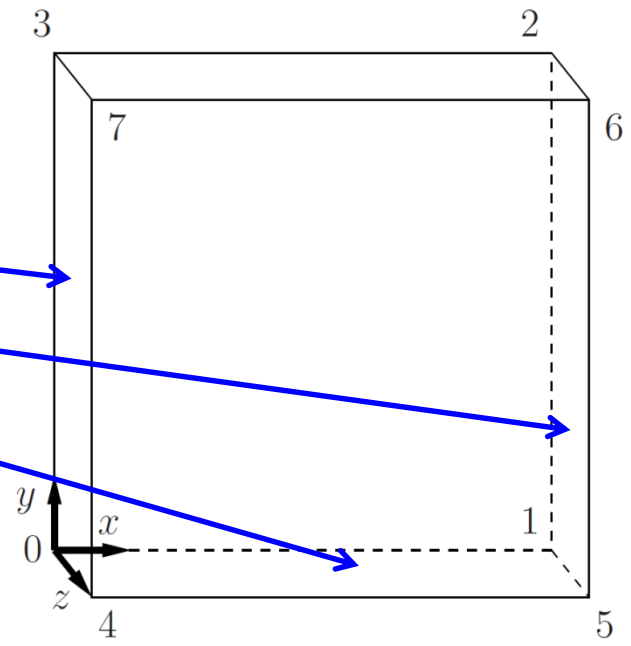
Edit blockMeshDict file (2)

```
40  boundary
41  (
42      movingWall
43      {
44          type wall;
45          faces
46          (
47              (3 7 6 2)
48          );
49      }
```



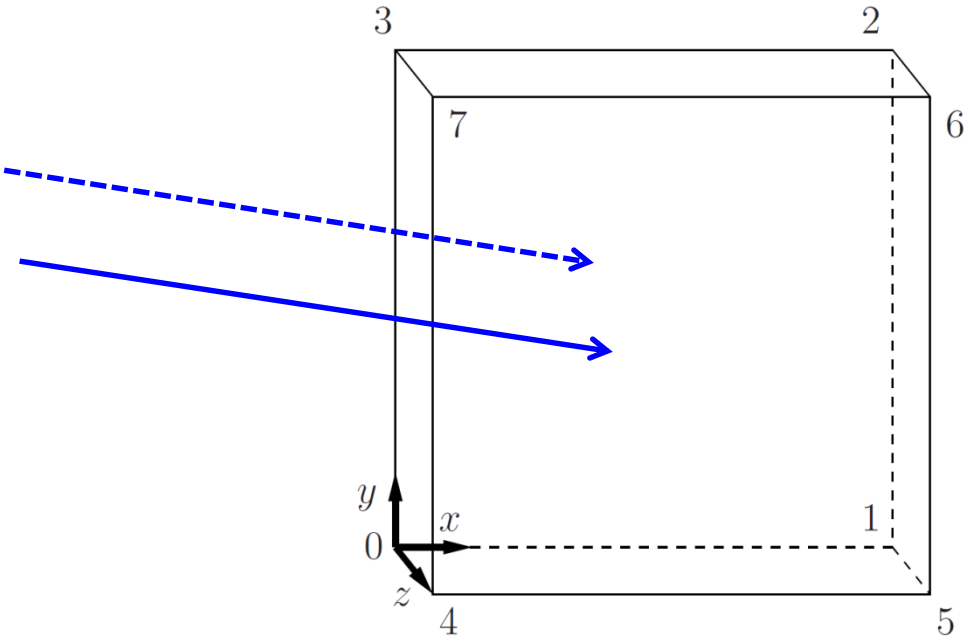
Edit blockMeshDict file (3)

```
50      fixedWalls
51      {
52          type wall;
53          faces
54          (
55              (0 4 7 3)
56              (2 6 5 1)
57              (1 5 4 0)
58          );
59      }
```



Edit blockMeshDict file (4)

```
53      frontAndBack
54      {
55          type empty;
56          faces
57          (
58              (0 3 2 1)
59              (4 5 6 7)
60          );
61  }
62  );
63
64  mergePatchPairs
65  (
66  );
```



Solver settings

- **constant directory also contains:**
  - File which defines physical/material properties, transportProperties
  - Files which define some other mesh properties, e.g. dynamicMeshDict
  - Files which defines turbulent properties RASProperties

Solver settings

- **dimensions/units in OpenFOAM**
  - Representation of SI system
    - `//dimensions [kg m sec K mol A cd ];`
    - `dimensions [0 2 -1 0 0 0 0];`
- **Note: for incompressible solvers it is not needed to specify density. Pressure is then represented as p/ρ**

- **transportProperties-representation of SI system**

```
transportModel Newtonian;//viscosity options: newtonian/non-newtonian
nu nu [ 0 2 -1 0 0 0 0 ] 0.01; // kinematic viscosity
```

Selected codes in icoFoam

```
// Refer to https://openfoamwiki.net/index.php/IcoFoam for more information
while (runTime.loop())
{
    Info<< "Time = " << runTime.timeName() << nl << endl;
    #include "readPISOControls.H"
    #include "CourantNo.H"
    fvVectorMatrix UEqn // Note the equation representation
    (
        fvm::ddt(U)
      + fvm::div(phi, U)
      - fvm::laplacian(nu, U)
    );
    solve(UEqn == -fvc::grad(p));
    // --- PISO loop
    for (int corr=0; corr<nCorr; corr++)
    {
        volScalarField rAU(1.0/UEqn.A());
    ...
    }
}
```

Solver settings

- **system directory contains:**
  - Files concerning solver parameters, as well as definition files for utility tools e.g. decomposeParDict
    - controlDict – simulation control and parameters, additional libraries to load and extra functions
    - fvSchemes – definition of discretization schemes
    - fvSolution – definitions of solver type, tolerances, relaxation factors

Solver settings-controlDict

- **controlDict: (basic time step control, how your results are written, etc.)**

```
application     icoFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         0.5;
deltaT          0.005;
writeControl    timeStep;
writeInterval   20;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
```

Solver settings-fvSchemes

- **fvSchemes:**

```
// time schemes (Euler , CrankNicholson, backward, steadyState )
ddtSchemes
{
    default         Euler;
}
// gradient schemes (Gauss , leastSquares, fourth, cellLimited, faceLimited )
gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
}
// convection and divergence schemes ( interpolation schemes used: linear, skewLinear,  cubicCorrected, upwind, linearUpwind, QUICK, TVD, SFCD, NVD)
divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
}
```

```
laplacianSchemes
{
    default         none;
    laplacian(nu,U) Gauss linear orthogonal;
    laplacian((1|A(U)),p) Gauss linear orthogonal;
}
```

Solver settings-fvSchemes

- **fvSchemes:**

```
// interpolation schemes to calculate
values on the faces (linear,
cubicCorrection, midPoint , upwind,
linearUpwind, skewLinear , QUICK, TVD,
limitedLinear , vanLeer , MUSCL,
limitedCubic, NVD, SFCD, Gamma )
interpolationSchemes
{
    default         linear;
    interpolate(HbyA) linear;
}
// schemes for surface normal gradient on
the faces ( corrected, uncorrected,
limited, bounded, fourth )
snGradSchemes
{
    default         orthogonal;
}
// lists the fields for which the flux is
generated in the application
fluxRequired
{
    default         no;
    p;
}
```

Solver settings-solution control

- **fvSolution:**

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0;
    }
}
// pressure - velocity coupling
// SIMPLE (Semi - Implicit Method for
Pressure - Linked Equations )
// PISO ( Pressure Implicit with Splitting
of Operators )
// PIMPLE ( Combination of SIMPLE and PISO
)
PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell          0;
    pRefValue       0;
}
```

http://www.openfoam.org/docs/user/fvSolution.php

Solver settings-time directory

- **Time directories contain field files (e.g. U, p, k, epsilon, omega, T etc.)**
- **Fields files store field solution values on all cells and boundary conditions on the computational domain**
- ***0* time directory is initial directory containing field files with initial field values and boundary conditions**
- **Common parts for all field files are:**
  - header
  - dimensions
  - internalField
  - boundaryField

Boundary Conditions (BCs)

- **base type (described purely in terms of geometry):**
  - `patch, wall, empty, symmetry, cyclic`
- **primitive type (base numerical patch condition assigned to a field variable on the patch):**
  - `fixedValue, fixedGradient, zeroGradient, mixed, directionMixed, calculated`
- **derived type (complex patch condition, derived from the primitive type, assigned to a field variable on the patch):**
  - `inletOutlet`

Initial and Boundary conditions: Velocity

- **U**

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{   movingWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    fixedWalls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    frontAndBack
    {
        type            empty;
    }
}
```

Initial and Boundary conditions: Pressure

- **p**

```
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform 0;

boundaryField
{
    movingWall
    {
        type            zeroGradient;
    }

    fixedWalls
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

system/decomposeParDict

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
numberOfSubdomains 16;
//Simple geometric decomposition in which the domain is split into
//pieces by direction, e.g. 4 pieces in x direction, 2 in y, 2 in z
method          simple;
simpleCoeffs
{
    n               ( 4 2 2 );
    delta           0.001;
}
```

Running cavity in parallel

- **decomposePar //specify the parallel run params**
- **mpirun --hostfile <machines> -np <nProcs>**
  <foamExec> <otherArgs> -parallel > log
  - Examples on Mike:

```
mpirun --hostfile $PBS_NODEFILE -np 16 icoFoam -parallel > log
```

  - Examples on QB2 (use proper values in decomposePar):

```
mpirun --hostfile $PBS_NODEFILE -np 20 icoFoam -parallel > log
```

- **reconstructPar //merge time directories sets from each processor**
- **See:**

```
/work/$USER/hpcengr/intro_of/cavity_parallel_is
/work/$USER/hpcengr/intro_of/cavity_parallel
```

Running cavity in parallel

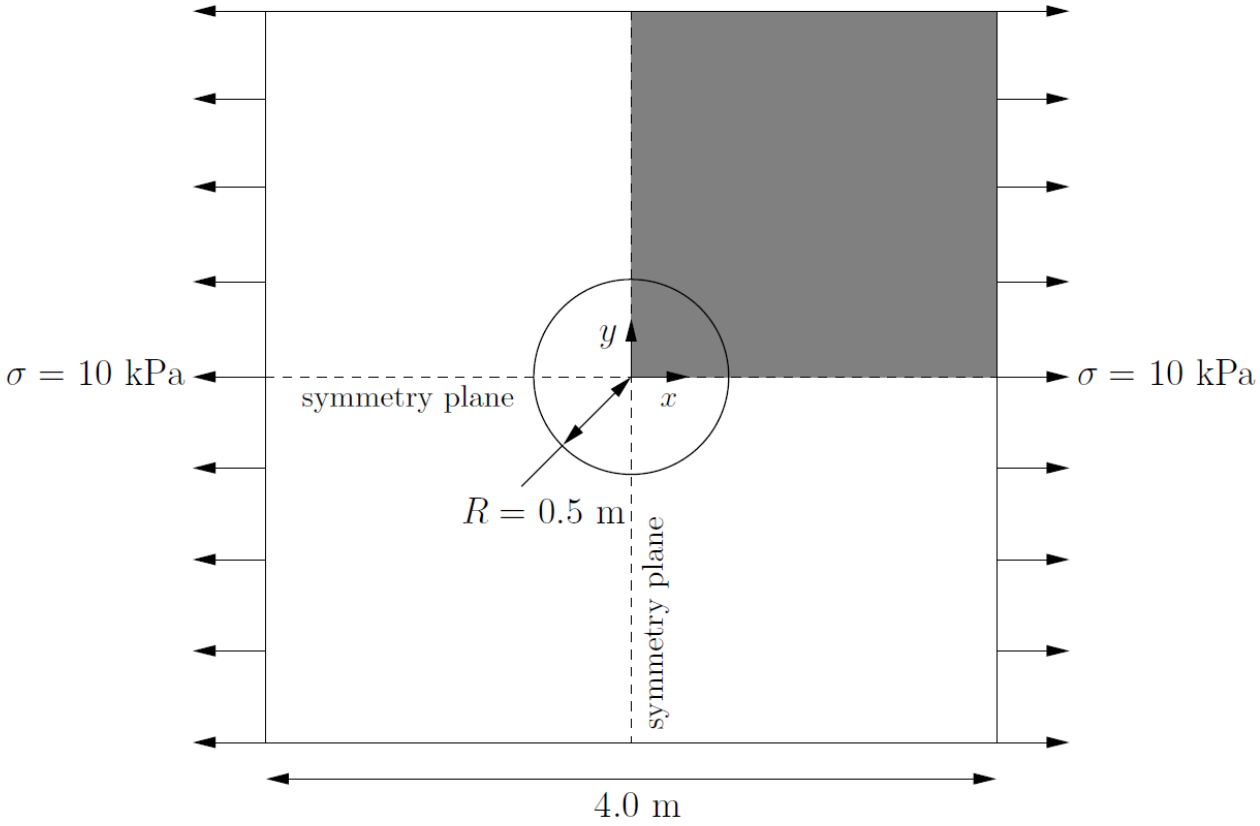
- **On interactive session:**

```
$ cd /work/$USER/hpcengr/intro_of/cavity_parallel_is
$ blockMesh
$ vi system/decomposeParDict
$ decomposePar   #break up the domain
$ mpirun --hostfile $PBS_NODEFILE -np 16 icoFoam -parallel
$ reconstructPar #combine results from domain directory (processor*)
```

- **Via batch mode:**

```
$ cd /work/$USER/hpcengr/intro_of
$ ./cavity_parallel_run.sh
```

Stress analysis of plateHole



Analytical Solution:

$$\left(\sigma_{xx}\right)_{x=0} = \begin{cases} \sigma\left(1+\dfrac{R^2}{2y^2}+\dfrac{3R^4}{2y^4}\right) & \text{for } |y| \geq R \\ 0 & \text{for } |y| < R \end{cases}$$

Part of the solidDisplacementFoam code

```
do // loop for residual and iterations
        {
            if (thermalStress)
            {
                volScalarField& T = Tptr();
                solve
                (
                    fvm::ddt(T) == fvm::laplacian(DT, T)
                );
            }

            {
                fvVectorMatrix DEqn // not a N-S equation
                (
                    fvm::d2dt2(D)
                 ==
                    fvm::laplacian(2*mu + lambda, D, "laplacian(DD,D)")
                  + divSigmaExp
                );
...
} while (initialResidual > convergenceTolerance && ++iCorr < nCorr);
```

Run stress analysis case

- **Steps of running plateHole on Mike:**

```
$ cd /work/$USER/foam_run/intro_of/plateHole
$ blockMesh #generate geometry
$ checkMesh #tool for checking the mesh quality
$ solidDisplacementFoam #running the stress analysis solver
$ foamToVTK #convert VTK format, optional
$ paraFoam  #post-processing
```

Post-processing

- **Most used post-processing software for OpenFOAM data visualization is *Paraview***
- **paraFoam – script for automatic import of OpenFOAM results into Paraview**
- **OpenFOAM Data transformation in other formats: e.g. foamToVTK (also used by Paraview)**

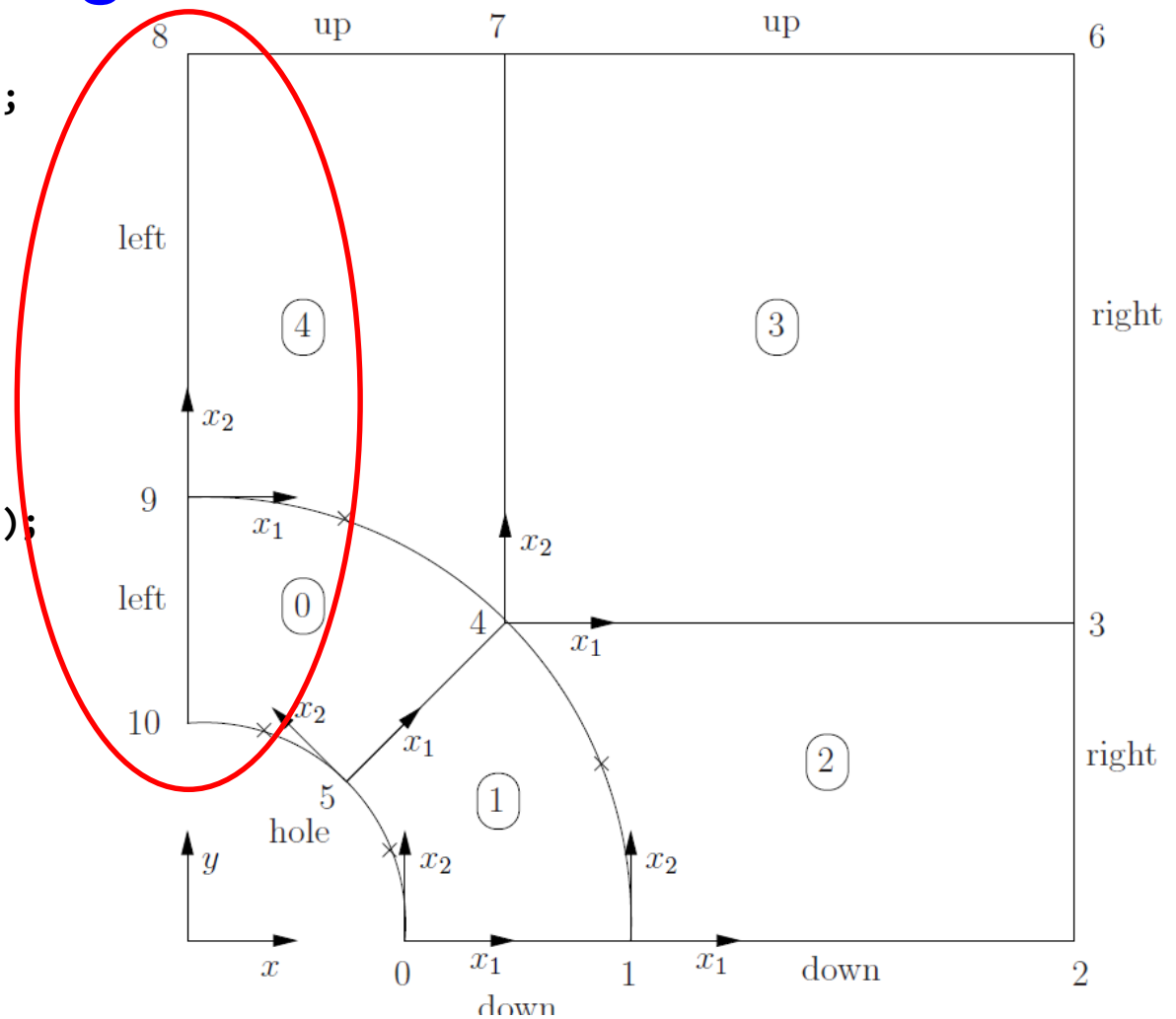
Post-processing

- ***sample*** **– utility used for sampling**
- **Sample setups are defined in system/sampleDict**
- **Sample data are stored in the new (automatically) created subdirectory sets**
- **Example:**

```
$ cd /work/$USER/foam_run/intro_of/plateHole
$ foamCalc components sigma #calculates new fields from existing ones.
$ sample
```

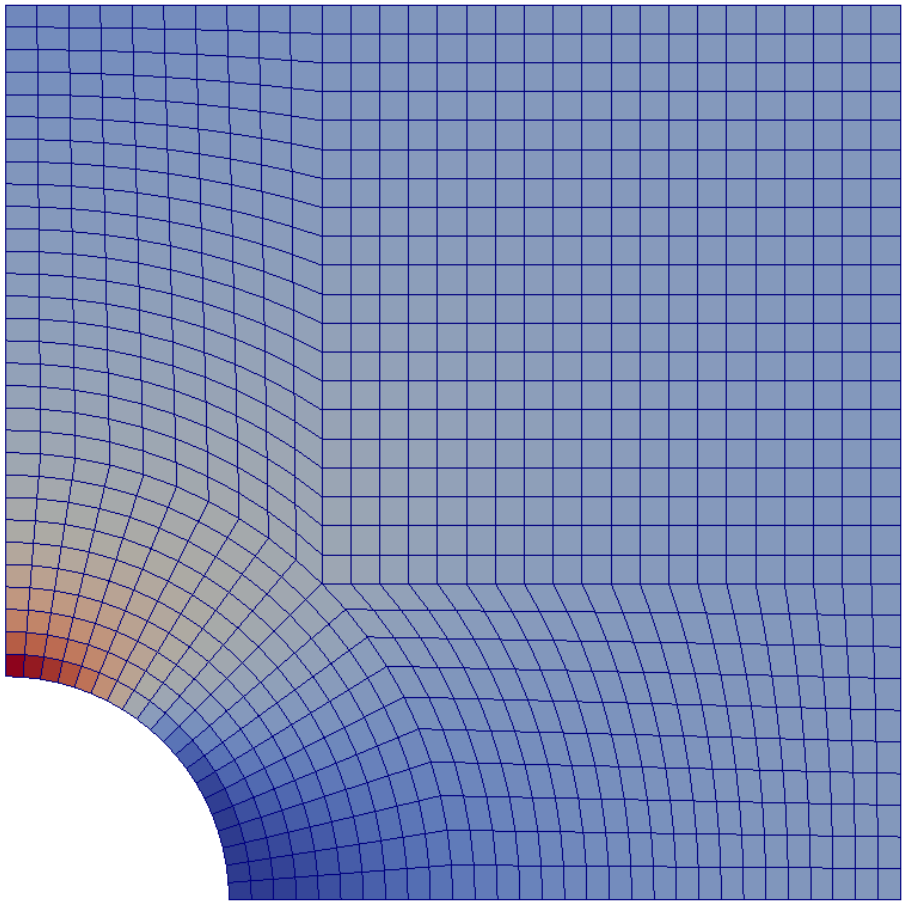
sampleDict for the plateHole case along the left line

```
/*OpenFOAM file header*/
interpolationScheme cellPoint;
setFormat       raw;
sets
(
    leftPatch
    {
        type    uniform;
        axis    y;
        start   ( 0 0.5 0.25 );
        end     ( 0 2 0.25 );
        nPoints 100;
    }
);
fields          ( sigmaxx );
```



sampleDict for the plateHole case for the entire surface

```
/*OpenFOAM file header*/
interpolationScheme cellPoint;
surfaceFormat   vtk;
surfaces
(
    sigmaxx
    {
        type plane;
        basePoint ( 0 0 0.25 );
        normalVector ( 0 0 1 );
    }
);
fields          ( sigmaxx );
```



Exercise

- **Run the cavity case and sample:**
  1. along middle-y axis
  2. surface

Develop your own solver

- **A simple commented overview of the icoFoam PISO solver, see http://openfoamwiki.net/index.php/IcoFoam**

```
// from the last solution of velocity, extract the diag. term from the matrix and store the reciprocal
// note that the matrix coefficients are functions of U due to the non-linearity of convection.
            volScalarField rUA = 1.0/UEqn.A();

// take a Jacobi pass and update U.  See Hrv Jasak's thesis eqn. 3.137 and Henrik Rusche's thesis, eqn. 2.43
// UEqn.H is the right-hand side of the UEqn minus the product of (the off-diagonal terms and U).
// Note that since the pressure gradient is not included in the UEqn. above, this gives us U without
// the pressure gradient.  Also note that UEqn.H() is a function of U.

            U = rUA*UEqn.H();

// calculate the fluxes by dotting the interpolated velocity (to cell faces) with face normals
// The ddtPhiCorr term accounts for the divergence of the face velocity field by taking out the
//  difference between the interpolated velocity and the flux.
            phi = (fvc::interpolate(U) & mesh.Sf())
                + fvc::ddtPhiCorr(rUA, U, phi);
```

- **Create your own solver by copy (on SuperMikeII): /usr/local/packages/OpenFOAM/2.2.1/Intel-13.0-openmpi-1.6.3/OpenFOAM-2.2.1/applications/solvers/incompressible/icoFoam**

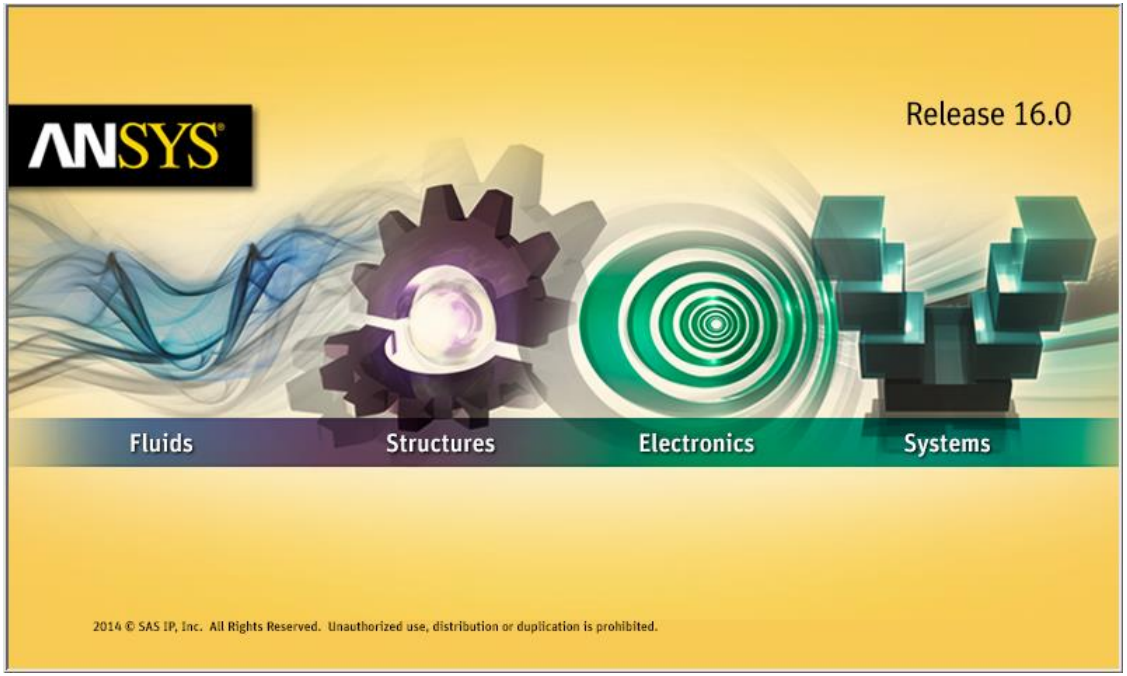
Documentation and Help

- **OpenFOAM course: http://www.tfd.chalmers.se/~hani/kurser/OS_CFD/**
- **OpenFOAM homepage: www.openfoam.com**
- **OpenFOAM User Guide, Programmers Guide, Doxygen**
- **OpenFOAM Wiki: www.openfoamwiki.net**
- **OpenFOAM-extend:**
  - http://sourceforge.net/projects/openfoam-extend/,
  - http://www.foam-extend.org
  - http://extend-project.de
- **OpenFOAM Forum: http://www.cfd-online.com/Forums/openfoam/**
- **OpenFOAM workshop: www.openfoamworkshop.org**
- **CoCoons project: http://www.cocoons-project.org/**

- **User forum:**
  - http://www.cfd-online.com/Forums/openfoam/

Brief Overview of Ansys

- **ANSYS is a general purpose software, used to simulate interactions of all disciplines of physics, structural, vibration, fluid dynamics, heat transfer and electromagnetic for engineers.**
- **Go through the ANSYS/FLUENT documentation carefully for detailed usage**



Example running Ansys Fluent

- **Add Ansys to you environment**
- **On SuperMike2, use softenv key:**

```
+ansys-16.0
```

- **On SuperMIC**

```
$ module av ansys
------- /usr/local/packages/Modules/modulefiles/apps -----------
ansys/15.0
$ module load ansys/15.0
```

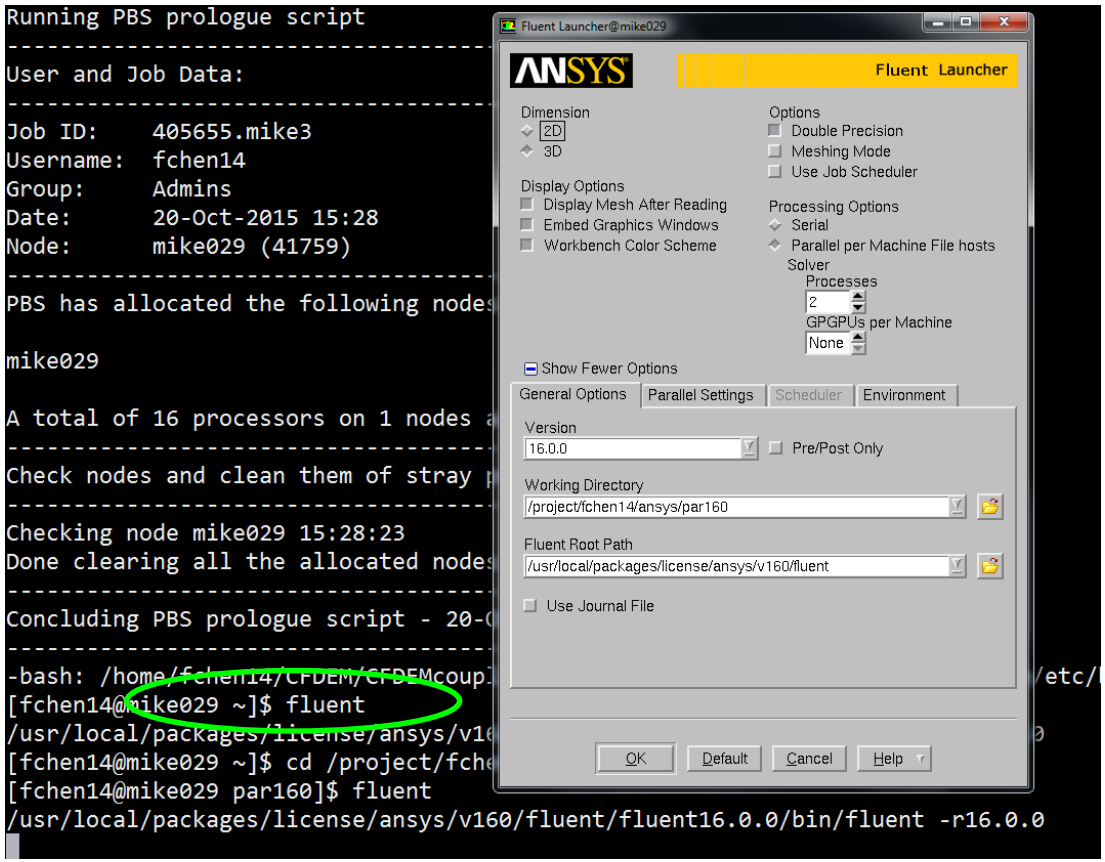
- **Setup/Verify your geometry and mesh settings in Fluent front end**
- **Write your job script (focus of the next few slides)**

Review Geometry using Fluent GUI (1)

- **Do *NOT* start ANSYS GUI on head node!!!**
- **Start an interactive session on SuperMike2:**

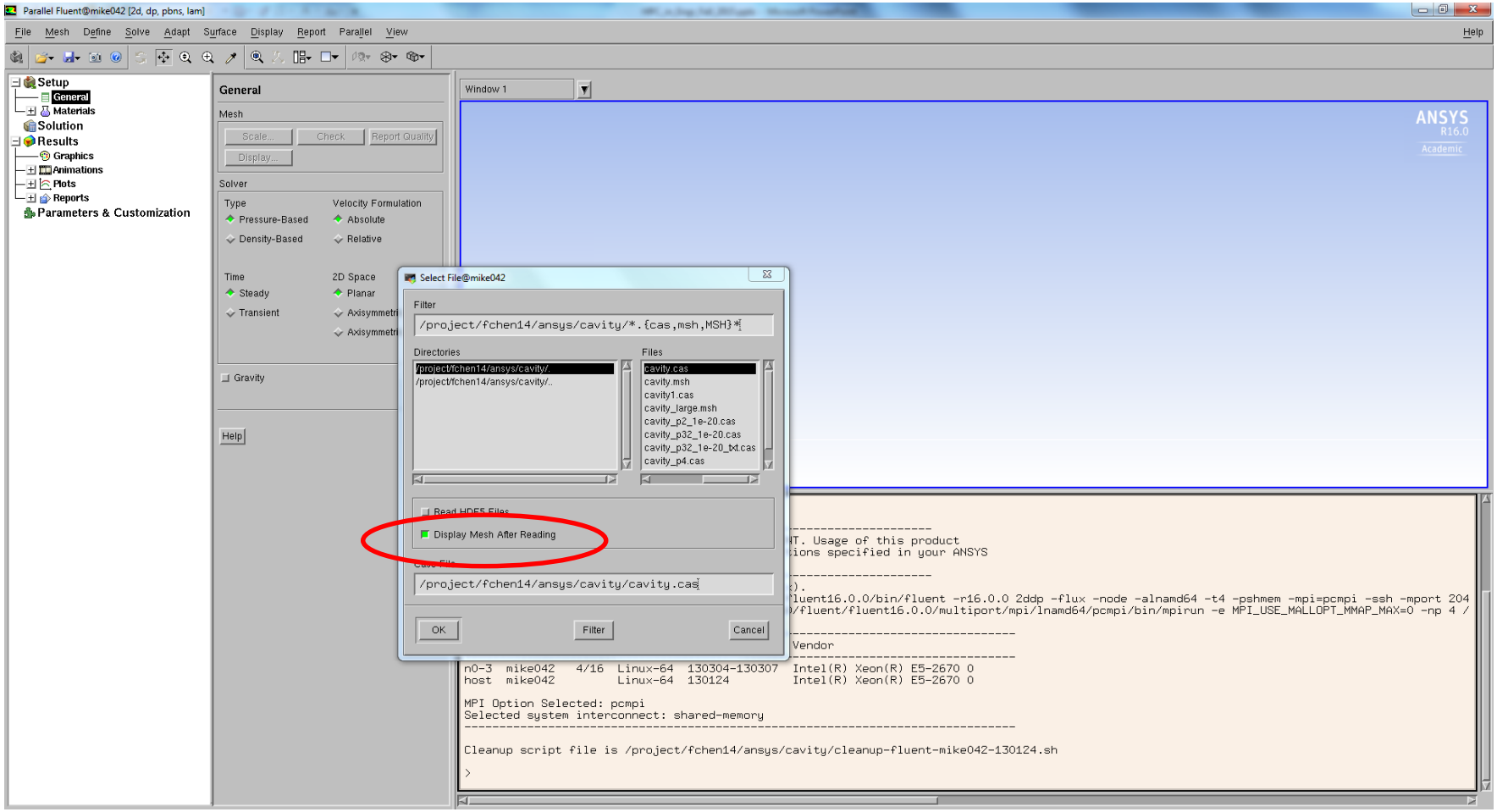
```
$ qsub -I -X -l nodes=1:ppn=16 -l walltime=01:00:00 -A your_allocation
```

- **Start the Ansys Fluent Launcher by typing `fluent` on terminal:**

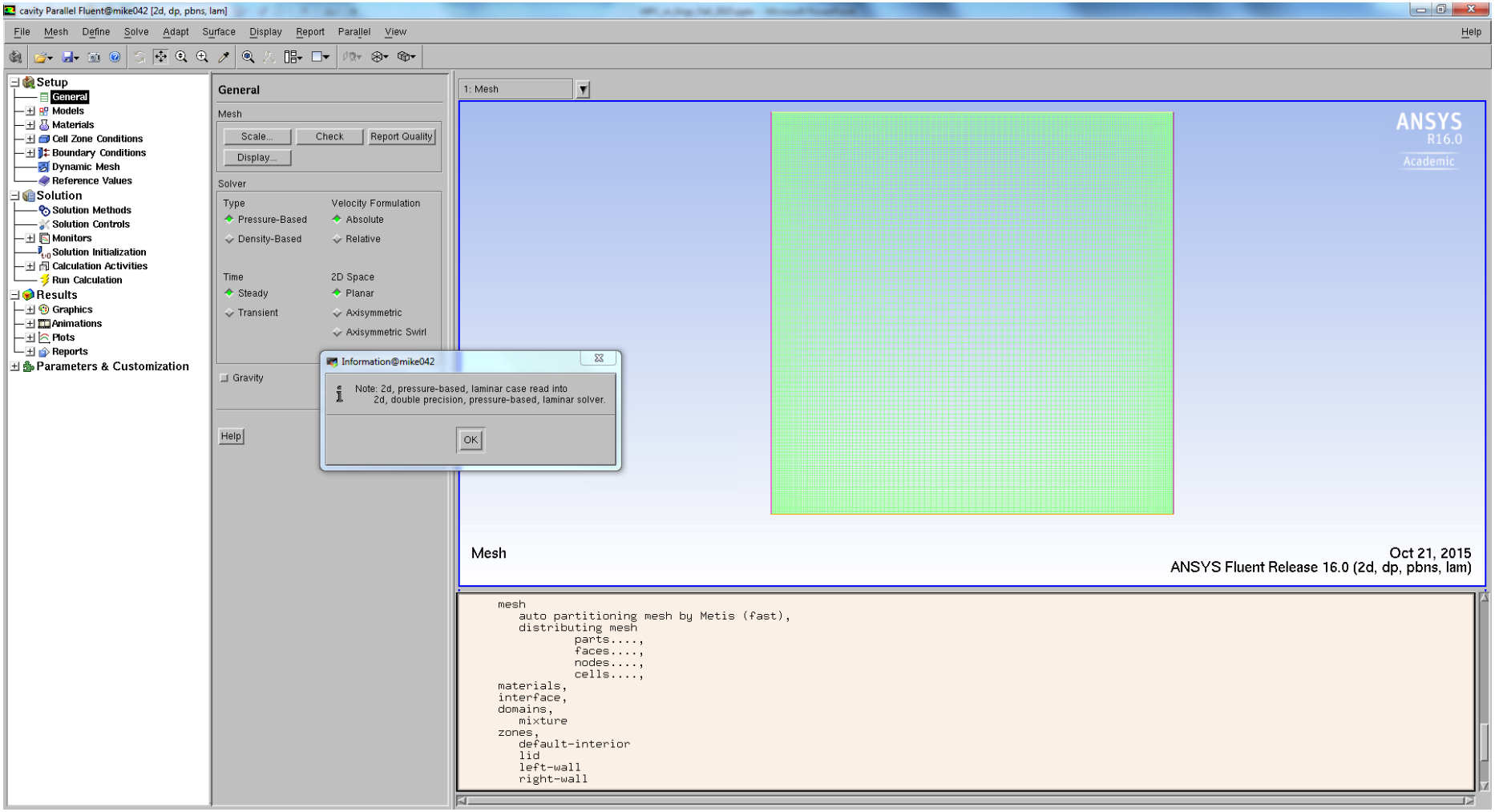


Review Geometry using Fluent GUI (2)

- Select `File > Read > Case` from the menu, select `cavity.cas`:
- Select "`Display Mesh After Reading`" option:

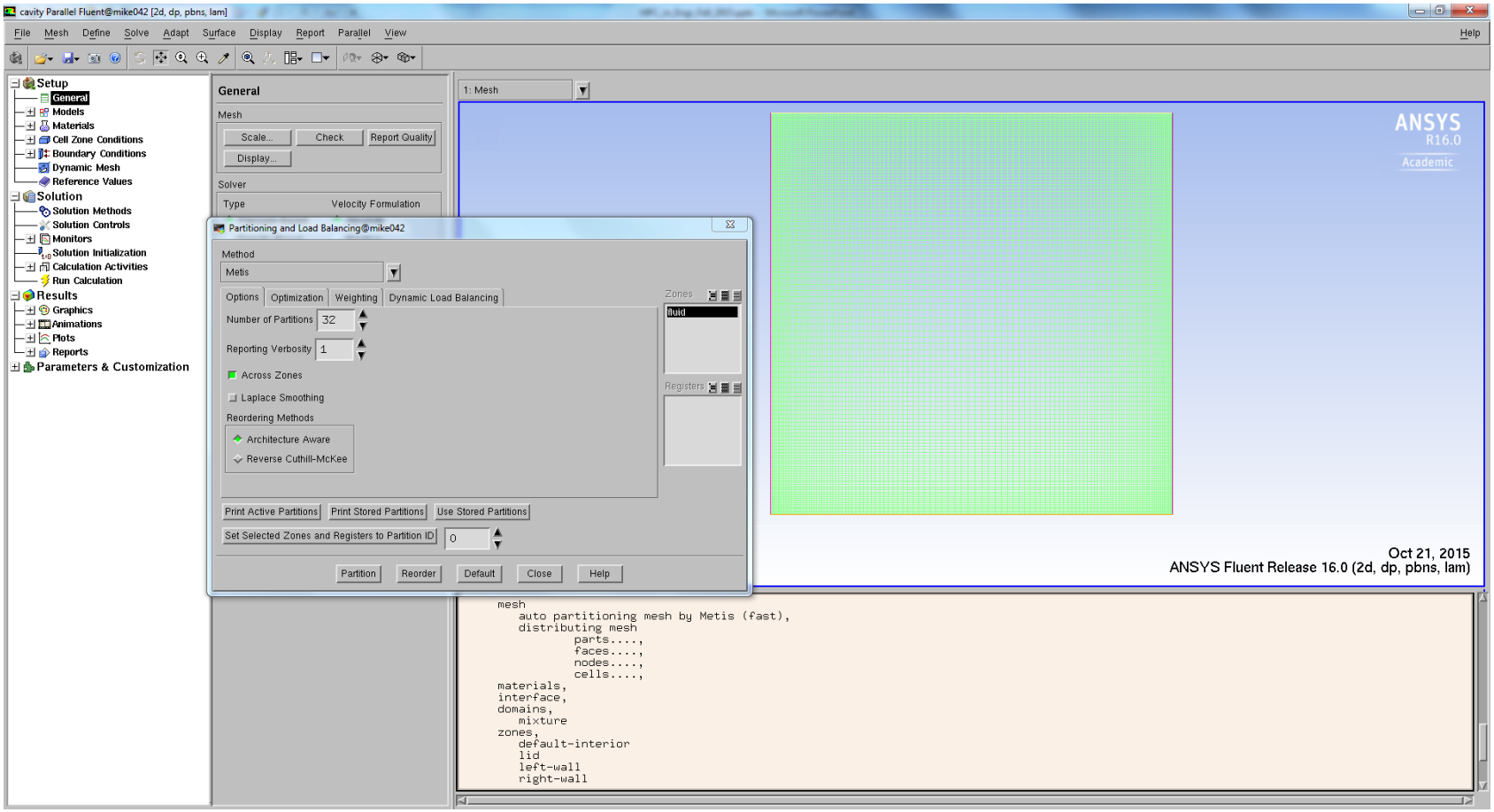


Review the 2D cavity Geometry



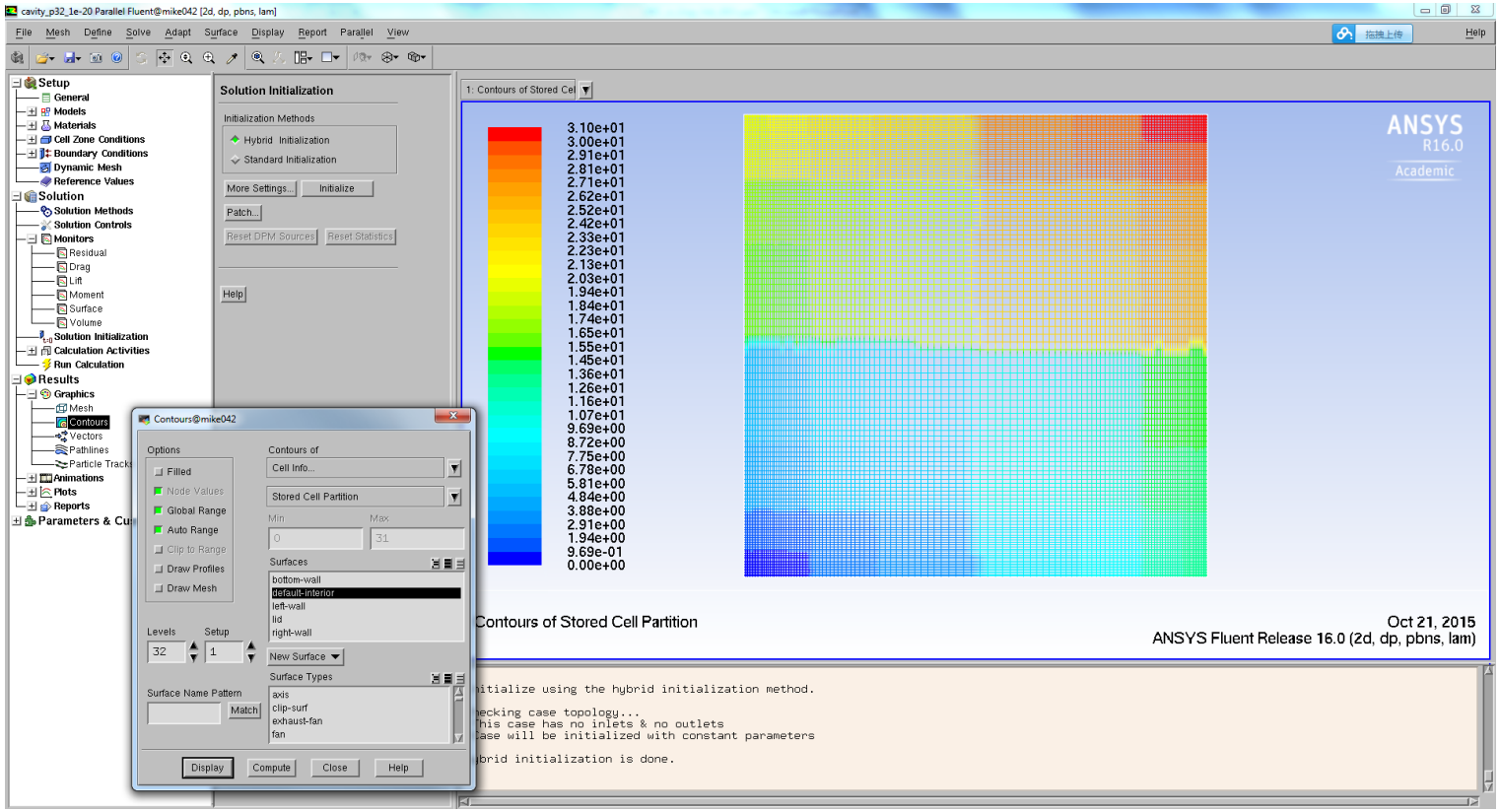
Partition the Existing Mesh

- **Select `Parallel > Partition and Load Balancing`**
  - Select **`Cartesian Axes`** under **`Methods`**
  - Set **`Number of Partitions`** to **`32`** in **`Options`** tab



Graphically Examine the Partitions

- **Initialize solution (required to visualize partitions) by:**
  - Solution > Solution Initialize > Initialize
- **Plot the partition contour by:**
  - Results > Contours > Edit: Set Levels to 32, Select default-interior under Surfaces



PBS Batch Job Script (2 nodes, 32 process)

```
#!/bin/bash
#PBS -l nodes=2:ppn=16
#PBS -l walltime=1:00:00
#PBS -A your_allocation_name
#PBS -q workq
#PBS -N par_cavity

cd $PBS_O_WORKDIR
echo "start fluent run"
TIC=`date +%s.%N`
# run on 32 processes on 2 nodes, -cnf=$PBS_NODEFILE is important!!!
fluent 2ddp -g -i cavity.jrn -t32 -cnf=$PBS_NODEFILE -pinfiniband -
mpi=intel -ssh
TOC=`date +%s.%N`
J1_TIME=`echo "$TOC - $TIC" | bc -l`
echo "end fluent run"
echo "simulation took=$J1_TIME sec"
```

The breakdown of the fluent command

```
fluent $version -g -i $journal -t$nCPU -p$ic -cnf=$hostfile -mpi=$mpi -ssh
```

Example:

```
fluent 3ddp -g -i input.jrn -t32 -pib -cnf=$PBS_NODEFILE -mpi=intel -ssh
```

- **`fluent` is ANSYS FLUENT command.**
- **`$version:` specifies the version for ANSYS FLUENT.**
  - `2d`:2-Dimension; `3d`: 3-Dimension
  - `dp`: double precision; `sp`: single precision
- **`-g:` runs without gui or graphics.**
- **`-t$nCPU:` specifies number of processor for parallel computing.**
- **`-p$ic:` specify interconnect between nodes; `<ic>={default|myri|ib}`.**
- **`-cnf=$hostfile:` specify the hosts file**
- **`-i$journal` reads the specified journal file.**
- **`-mpi=<mpi>:` specify MPI implementation; `<mpi>={pcmpi | intel | ...}`**
- **For a more detailed reference use: `fluent -h`**

Ansys CAS file cavity.cas:

```
(0 "fluent16.0.0  build-id: 10427")
(0 "Machine Config:")
(4 (60 0 0 1 2 4 8 4 8 8 8 4))
(0 "Dimensions:")
(2 2)
(0 "Variables:")
(37 (
(sizing-function/max-resolution 60)
(sizing-function/resolution 3)
(dynamesh/collapse/skew-max 0.7)
(smooth-mesh/min-skew 0.4)
(adapt/coarsen/marker/visible? #f)
(adapt/coarsen/cell/visible? #t)
(adapt/coarsen/cell/filled? #t)
(adapt/refine/marker/visible? #f)
(adapt/refine/cell/visible? #t)
(adapt/refine/cell/filled? #t)
...
```

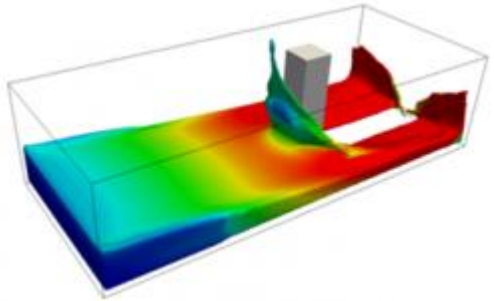
Ansys Journal File: “cavity.jrn”

- **Ansys journal file contains a list of instructions to pass to FLUENT, similar to a macro.**

```
; Read case file
rc cavity_p32.cas
; Initialize the solution
/solve/init/init
; Calculate 10k iterations
it 10000
; Write data file
wd it1k-cavity.dat
; say OK when asking for overwrite previous file
OK
; Exit Fluent
exit
; say yes when confirming
yes
```

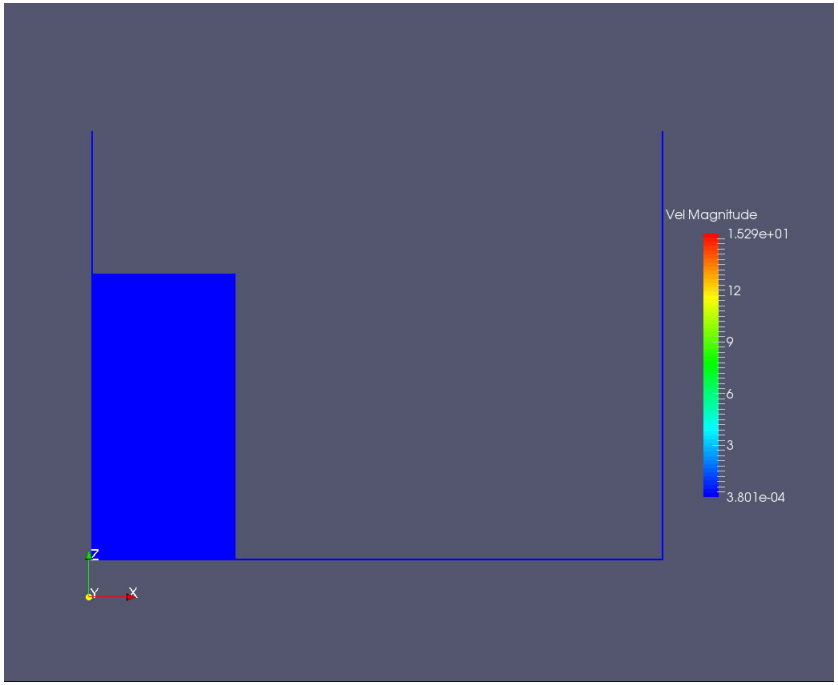
Example Running DualSPhysics

- **What is DualSPhysics**
  - DualSPHysics is based on the Smoothed Particle Hydrodynamics model named SPHysics (www.sphysics.org).
  - The code is developed to study free-surface flow phenomena where Eulerian methods can be difficult to apply, such as waves or impact of dam-breaks on off-shore structures.
- **Important HPC features**
  - DualSPHysics is a set of C++, CUDA and Java codes designed to deal with real-life engineering problems to carry out simulations on the CPU and GPU respectively.
  - Lagrangian Navier-Stokes fluid solver
  - **Utilizes GPU Accelerator**

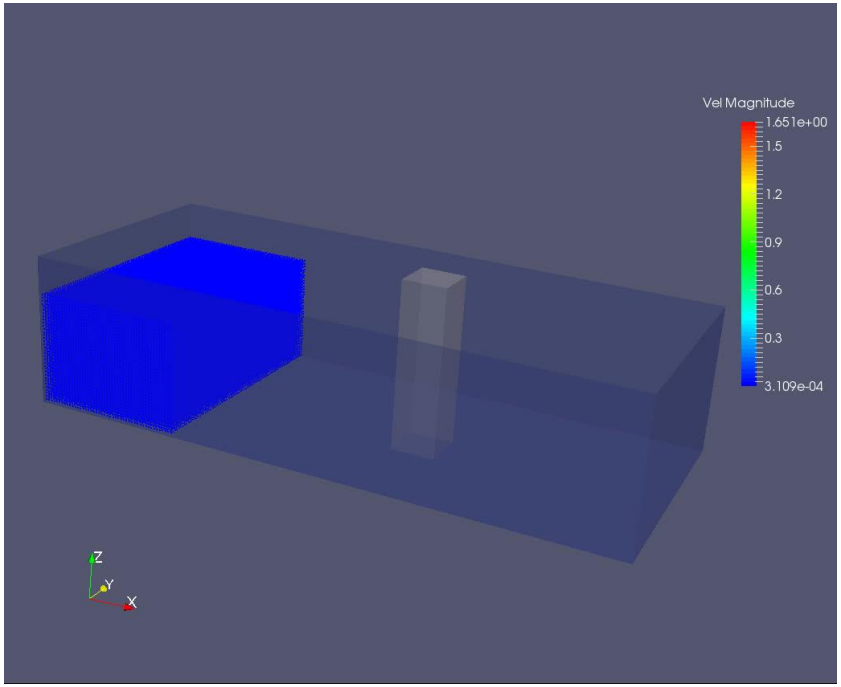


Animation of 2D/3D DamBreak with SPH

2D Dambreak

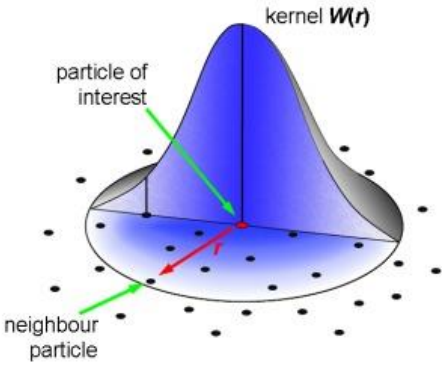


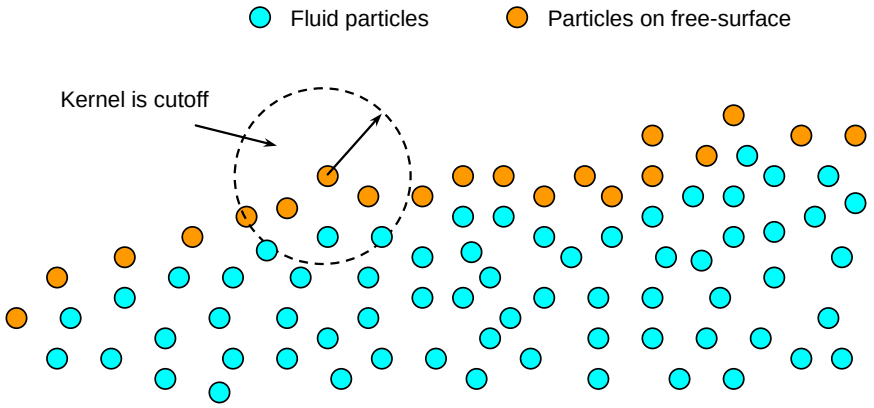
3D Dambreak



Some Theoretical Background

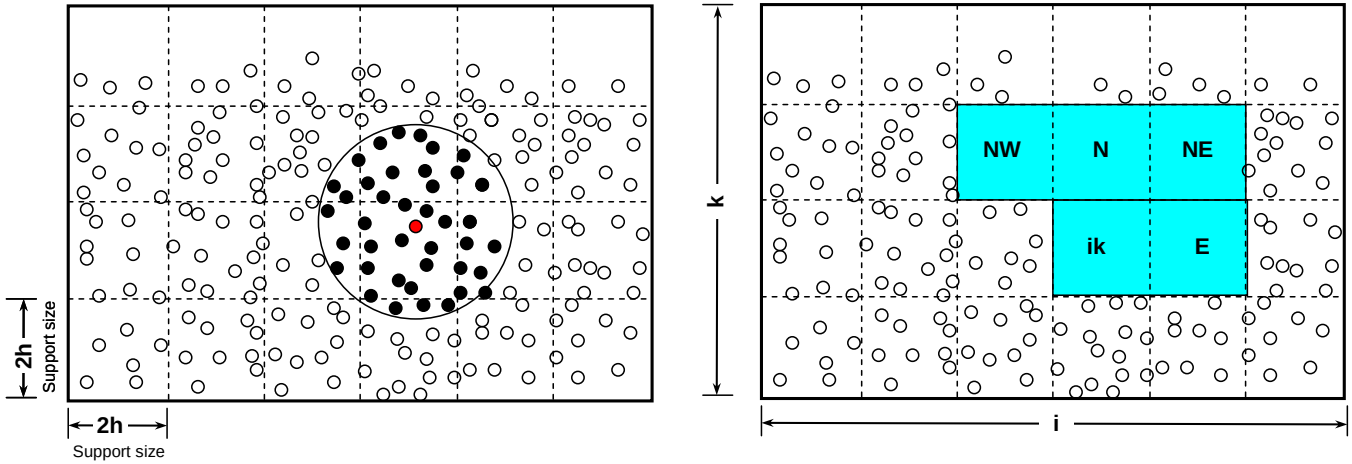
- **SPHysics is an open-source Smoothed Particle Hydrodynamics (SPH) model to solve Navier - Stokes Equations**
- **SPH is a Lagrangian meshless method that has been used in an expanding range of applications within the field of Computation Fluid Dynamics where particles represent the flow.**
- **Lagrangian method models can typically achieve high speedups with the usage of CUDA.**
  - The parallel power computing of GPUs are also applied for in DualSPysics methods where the same loops for each particle during the simulation can be parallelized.





Sub-steps of an SPH simulation iteration

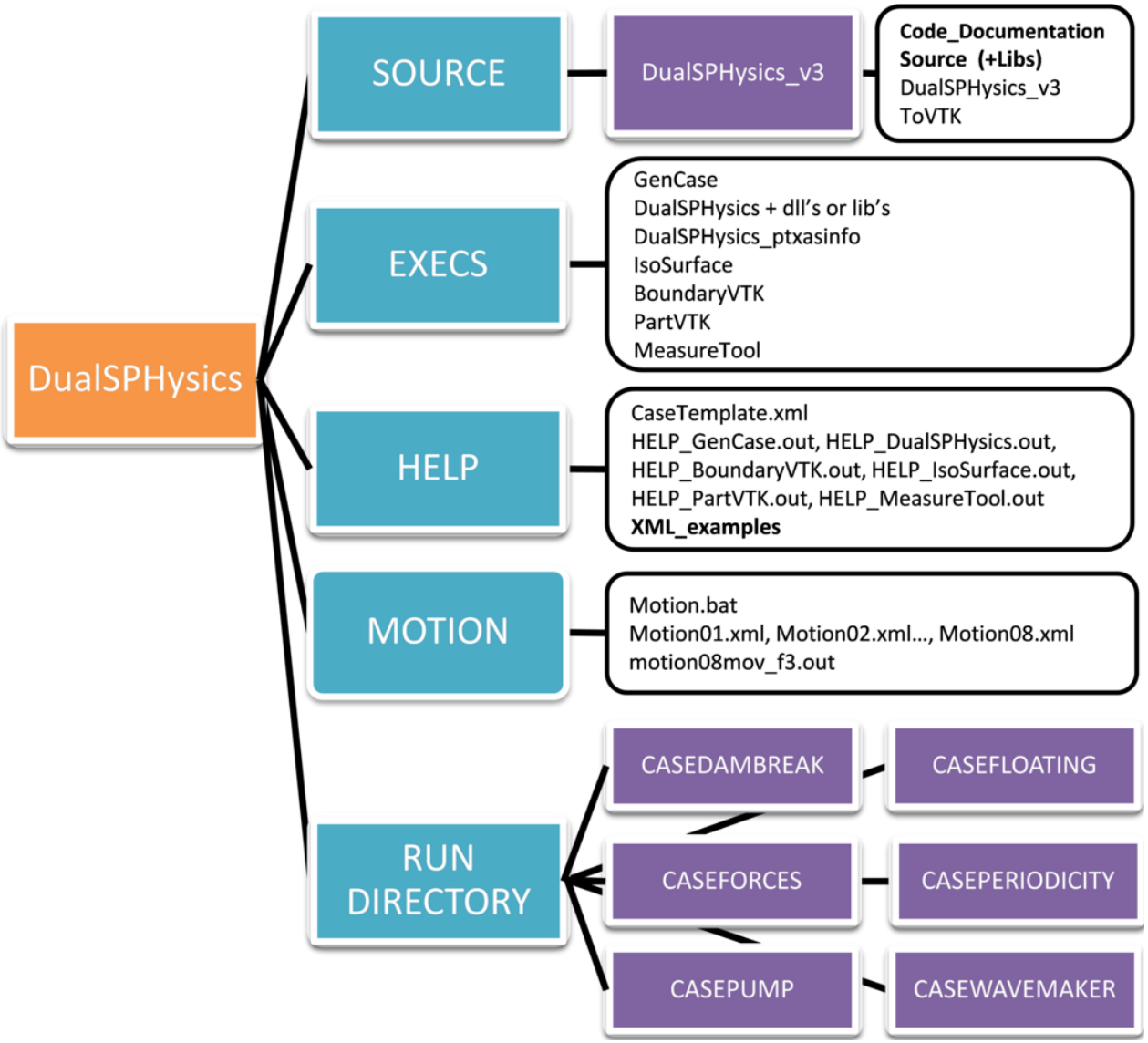
- **Neighbor list (Cell-linked list)**
  - Domain is divided in square cells of side 2h (or the size of the kernel domain) to create arrays list of particles.
- **Force computation**
  - Particles of the same cell and adjacent cells are candidates to be neighbors.
  - Each particle interacts with all its neighboring particles (d < 2h).
- **System Update (Most time consuming part, GPU implementation)**
  - Physical quantities are updated using the present time step physical values after time step is computed



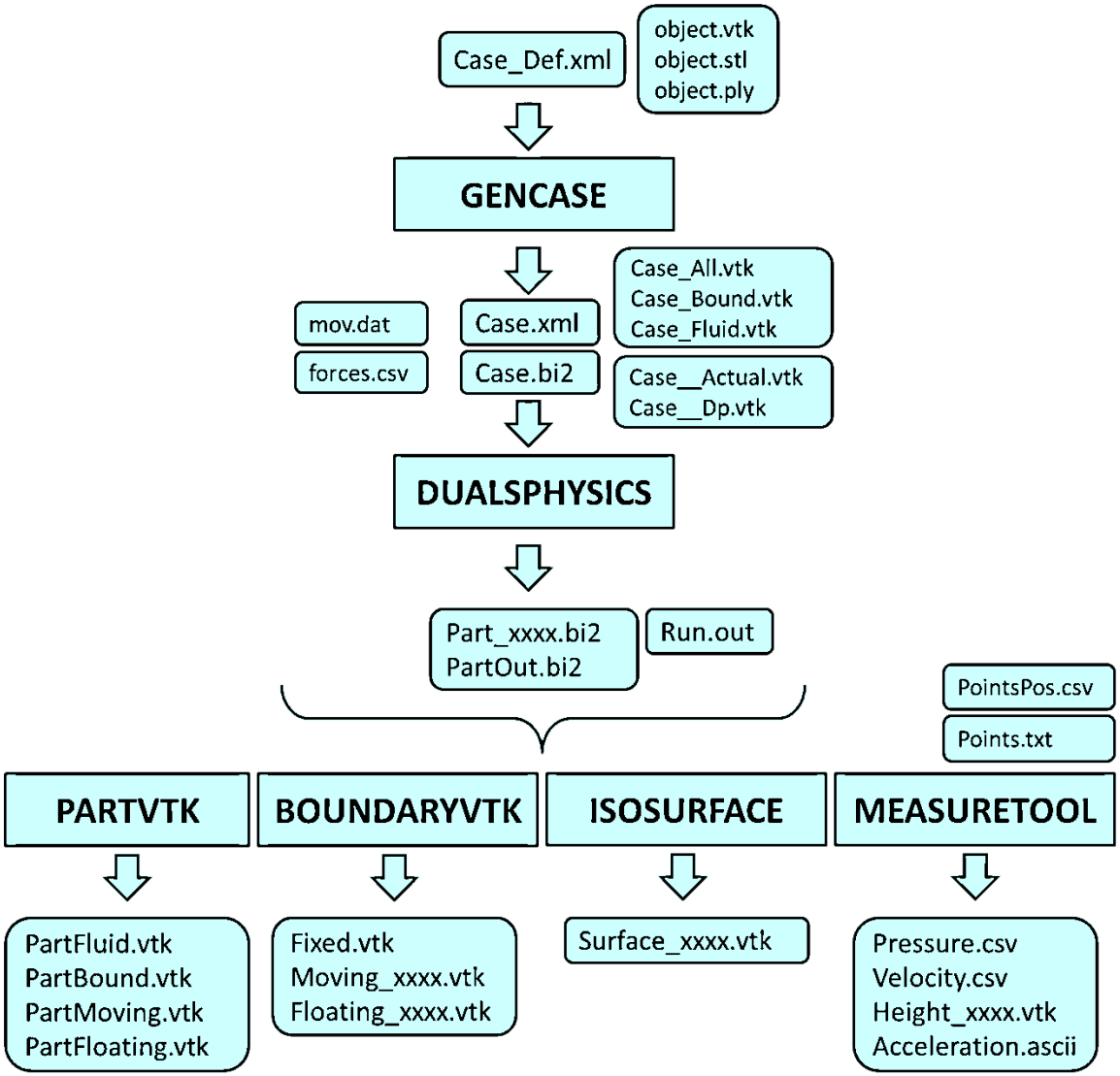
Compile and Install (on QB2)

```
$  tar zxf DualSPHysics_v3.2_Linux_x64.tar.gz
$  cd DualSPHysics_v3.2_Linux_x64.tar.gz
$  #change cuda library and compiler in the Makefile
$  make
icpc -c -c -O3 -openmp -D_WITHGPU -march=native -I./ -
I/usr/local/packages/cuda/6.0//include main.cpp
icpc -c -c -O3 -openmp -D_WITHGPU -march=native -I./ -
I/usr/local/packages/cuda/6.0//include Functions.cpp
...
rm -rf *.o
  --- Compiled Release GPU version ---
```

Directory Tree of DualSPhysics Code



DualSPhysics Workflow



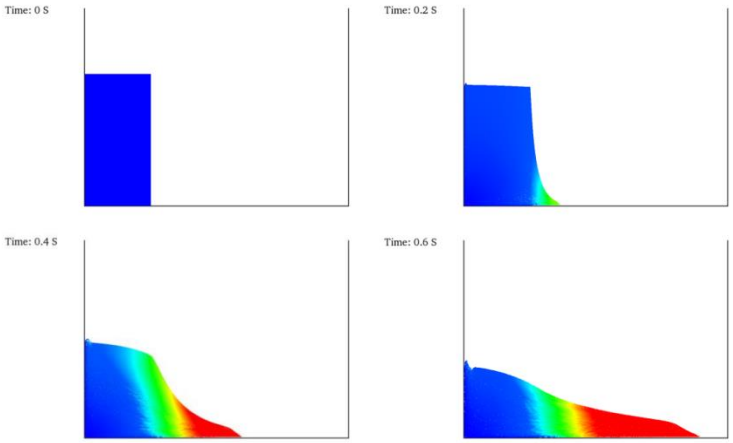
Run CPU/GPU based DualSPhysics models

- **Request an interactive session to get a compute node and then run the following commands:**

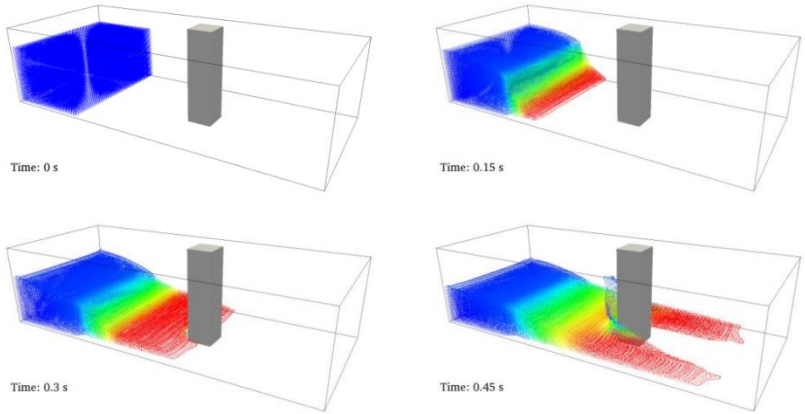
```
cd /project/fchen14/DualSPHysics_v3.2_Linux_x64/RUN_DIRECTORY/CASEDAMBREAK
./CaseDambreak_linux64_CPU.sh > CaseDambreak_linux64_CPU.log
./CaseDambreak_linux64_GPU.sh > CaseDambreak_linux64_GPU.log
./CaseDambreakVal2D_linux64_CPU.sh > CaseDambreakVal2D_linux64_CPU.log
./CaseDambreakVal2D_linux64_GPU.sh > CaseDambreakVal2D_linux64_GPU.log
```

Performance Comparison for 2D/3D Dambreak Case on QB2

2D Dambreak



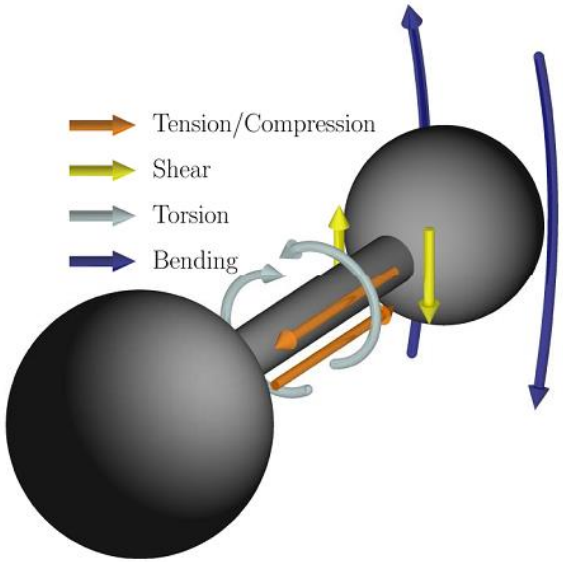
3D Dambreak



Dambreak Case	Simulation Time (sec)	Speedup
2D with CPU	136.9	--
2D with GPU	33.9	4.04x
3D with CPU	1405.1	--
3D with GPU	207.1	6.78x

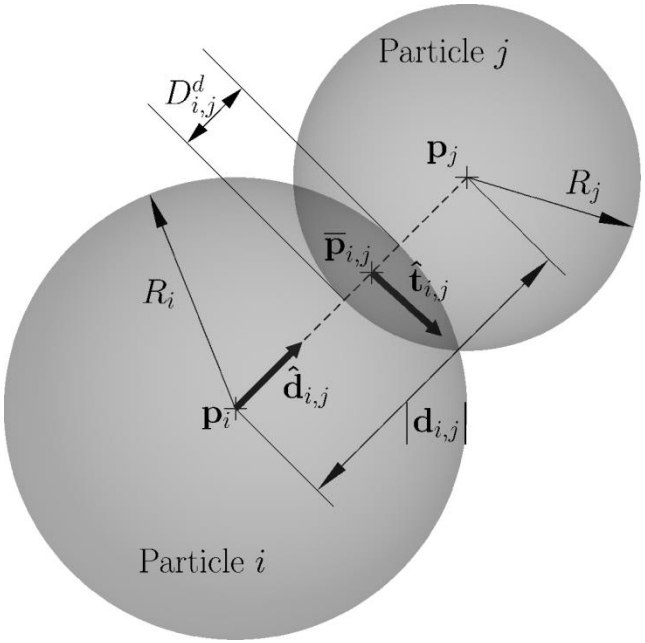
Discrete Element Code - LIGGGHTS

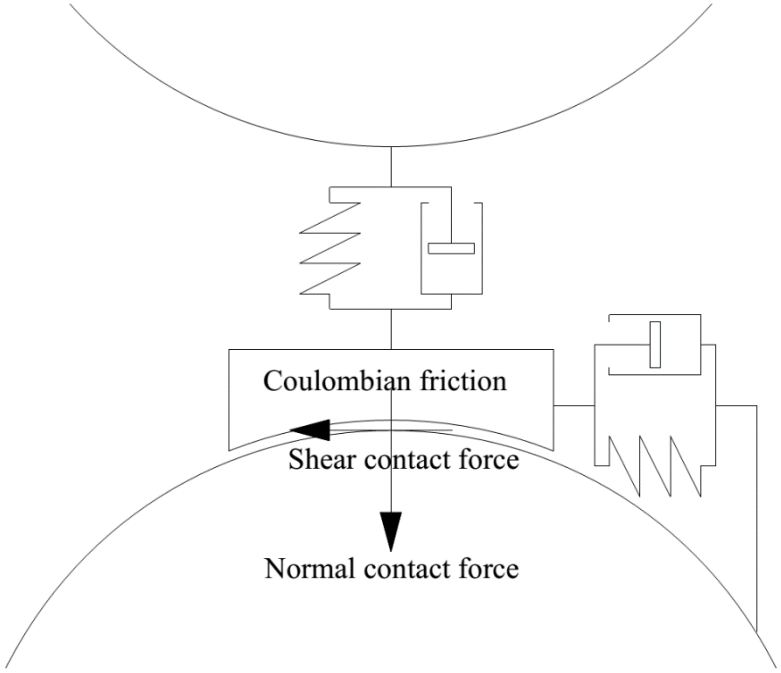
- **For granular material, PDEs based on continuum theory no longer apply.**
- **LIGGGHTS is an Open Source Discrete Element Method Particle Simulation Software based on LAMMPS.**
- **LIGGGHTS stands for LAMMPS Improved for General Granular and Granular Heat Transfer Simulations.**



Basic assumptions of DEM

- **Granular media (soil, rock) is considered to be composed of discrete particles instead of continuum.**
- **All particles are assumed to be rigid (No self-particle deformation)**
- **Inter-particle deformation is approximated by small overlapping.**





LIGGGHTS Features

- **Import and handling of complex geometries: STL walls and VTK tet volume meshes**
- **Moving mesh feature with a varierty of motion schemes and a model for conveyor belts**
- **Force and wear analysis on meshes as well as stress-controlled walls**
- **A variety of particle-particle contact implementations, including models for tangential history, non-spericity and cohesion**
- **Heat conduction between particles**
- **Particle insertion based on pre-defined volumes, meshes and particle streams from faces as well as particle growth and shrinkage**
- **Flexible definition of particle distributions**
- **Coupled module with OpenFOAM: CFDEM®coupling for CFD-DEM simulations and Lagrange-Euler coupling, which will be detailed later.**

Compile and Install LIGGGHTS

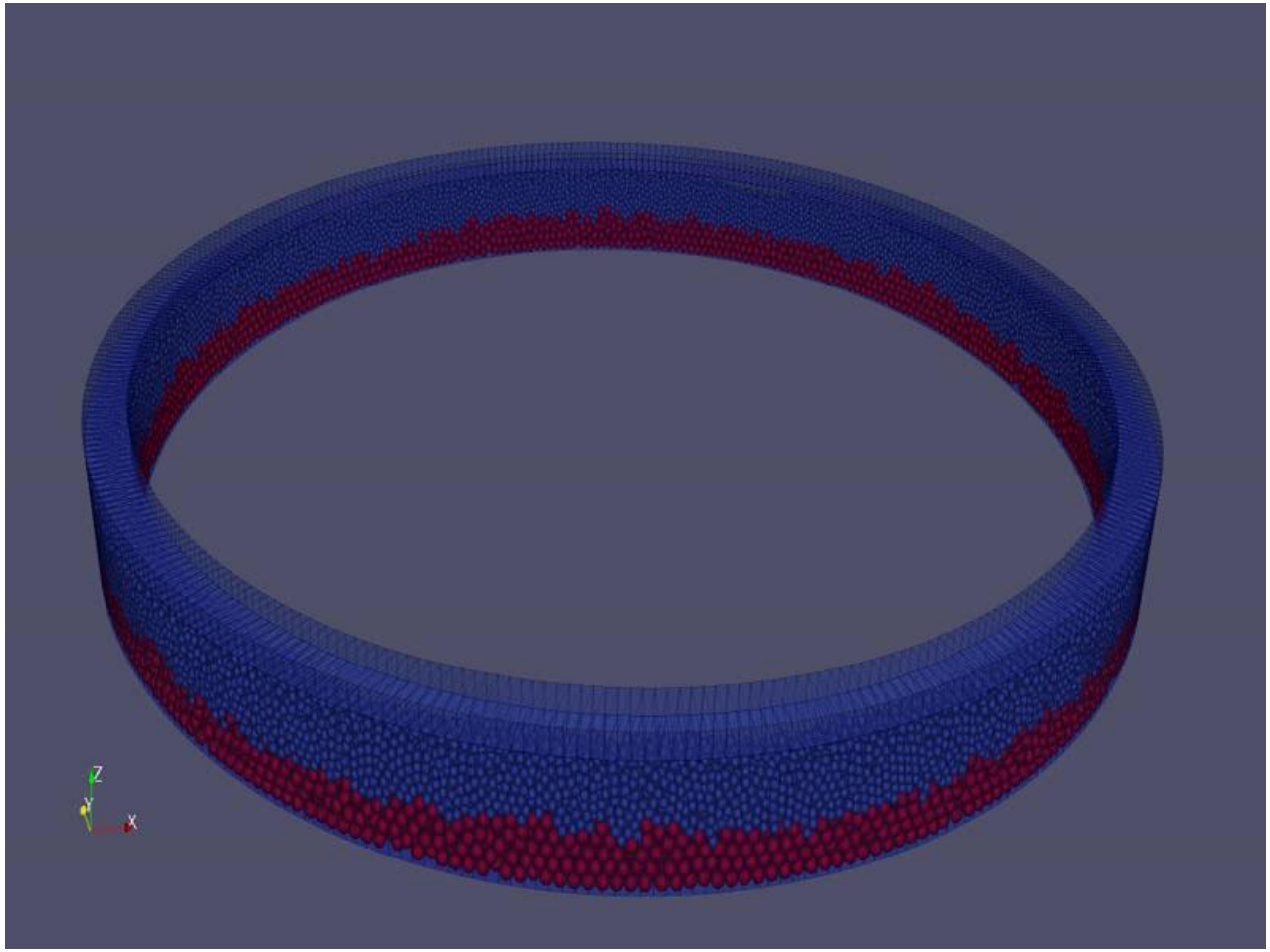
- **The compilation and installation is similar to LAMMPS.**
- **To clone the public LIGGGHTS(R)TUG repository, execute:**

```
$ git clone git@github.com:CFDEMproject/LIGGGHTS-PUBLIC.git
$ cd LIGGGHTS-PUBLIC/src
$ make fedora
$ ln -s lmp_fedora liggghts
```

- **This will make the MPI version of LIGGGHTS**
- **To run LIGGGHTS, use the following command:**

```
$ mpirun -np $NPROCS -f $PBS_NODEFILE `which liggghts` -in in.script
```

Example Problem: shearcell



Shearcell LIGGGHTS Input Script Segment

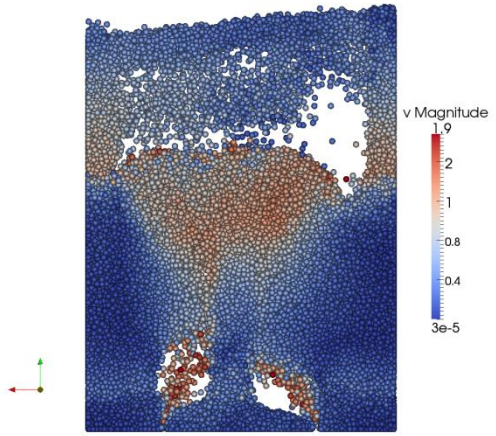
```
# Shear Cell example
### Initialization
echo both
units si
atom_style sphere
boundary f f f
newton off
communicate single vel yes
processors          4 5 1
# Declare domain
region reg block -0.325 0.325 -0.325 0.325 -0.001 0.081 units box
create_box 2 reg
...
dump dmp all custom 50000 res/dump*.lgts id type type x y z ix iy iz vx vy vz fx fy fz
&
omegax omegay omegaz radius
...
# Describe shearing action
fix movecad all move/mesh mesh cad2 rotate origin 0. 0. 0. axis 0. 0. 1. period 20.0
# Shear
run 30000000
```

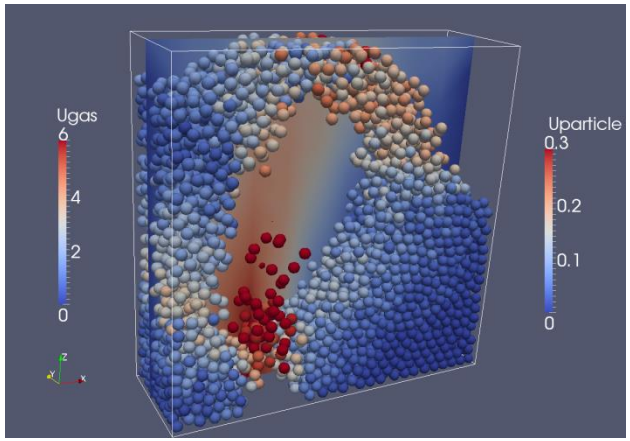
PBS script for shearcell

```
#!/bin/bash
#PBS -q workq
#PBS -A loni_loniadmin1
#PBS -l nodes=1:ppn=20
#PBS -l walltime=72:00:00
#PBS -j oe
#PBS -N shearcell

NPROCS=$(wc -l < $PBS_NODEFILE)
SRC_DIR=~/sedimentation/shearcell
WORK_DIR=/work/fchen14/shearcell
cp $SRC_DIR/*.stl $SRC_DIR/shearcell.liggghts.v320 $WORK_DIR

cd $WORK_DIR
mkdir -p $WORK_DIR/res
mpirun -np $NPROCS -machinefile $PBS_NODEFILE /project/fchen14/LIGGGHTS-
PUBLIC/src/lmp_fedora -in shearcell.liggghts.v320
# postprocessing particle data
cd $WORK_DIR/res
python $CFDEM_LPP_DIR/lpp.py dump*.lgts
```

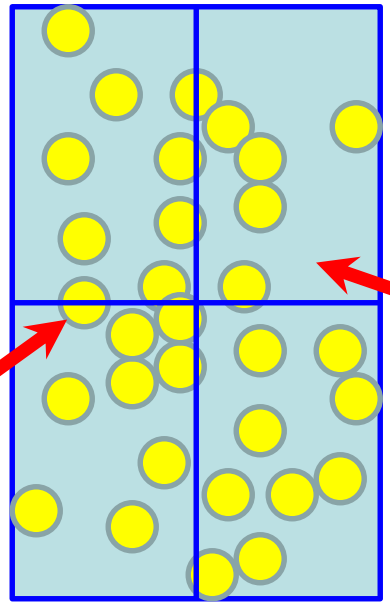
Coupled CFD-DEM methods

- **The CFD-DEM Method is a synthesis of CFD (FVM) and DEM to model coupled fluid-granular.**
- **The motion of the particles is resolved with DEM, and the CFD method is used to calculate the fluid flow.**
- **The granular phase which occupies a certain volume in each computational cell, is accounted for by introducing a "volume fraction" into the Navier-Stokes equations. Furthermore, the granular and fluid phase can exchange momentum, heat and mass.**
- **As FVM is an Eulerian method, DEM is a Lagrangian method, this CFD-DEM is a coupled Eulerian-Lagrangian method.**





Coupled DEM and FVM



Particle motion solved by DEM

Fluid motion by solving the Navier-Stokes Equation (FVM)

Fluid-Particle interaction by coupling DEM-FVM

Typical Applications of CFD-DEM

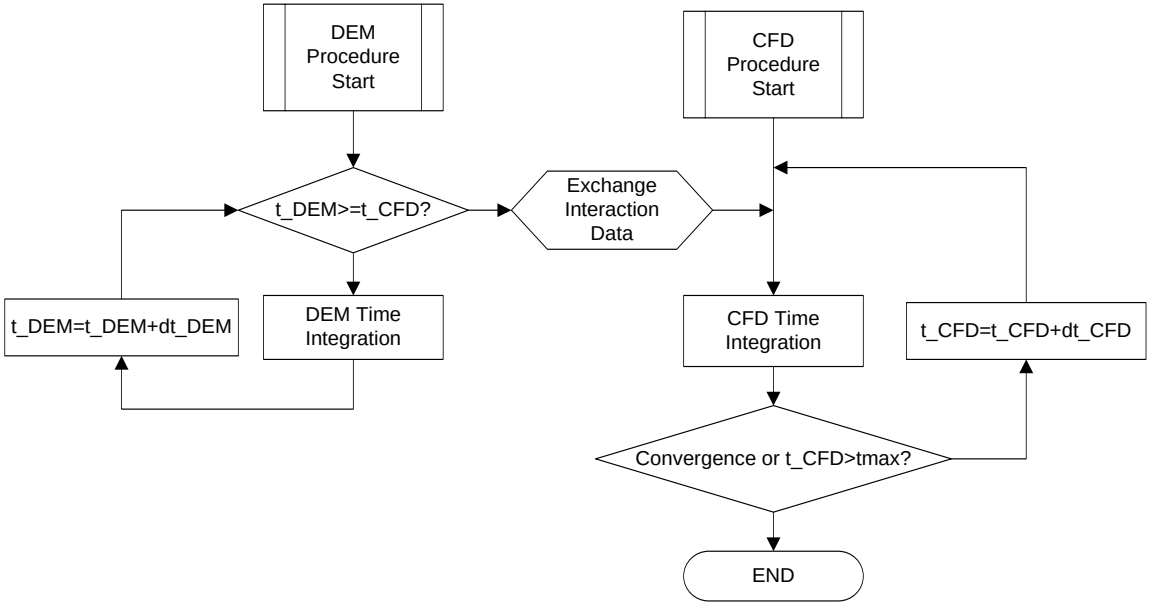
- **Fluidized beds**
- **Metallurgical processes (e.g. blast furnace)**
- **Pneumatic conveying**
- **Chemical and pharmaceutical reactors**
- **Any kind of granular flow influenced by the interstitial fluid phase**

LIGGGHTS Coupled with OpenFOAM

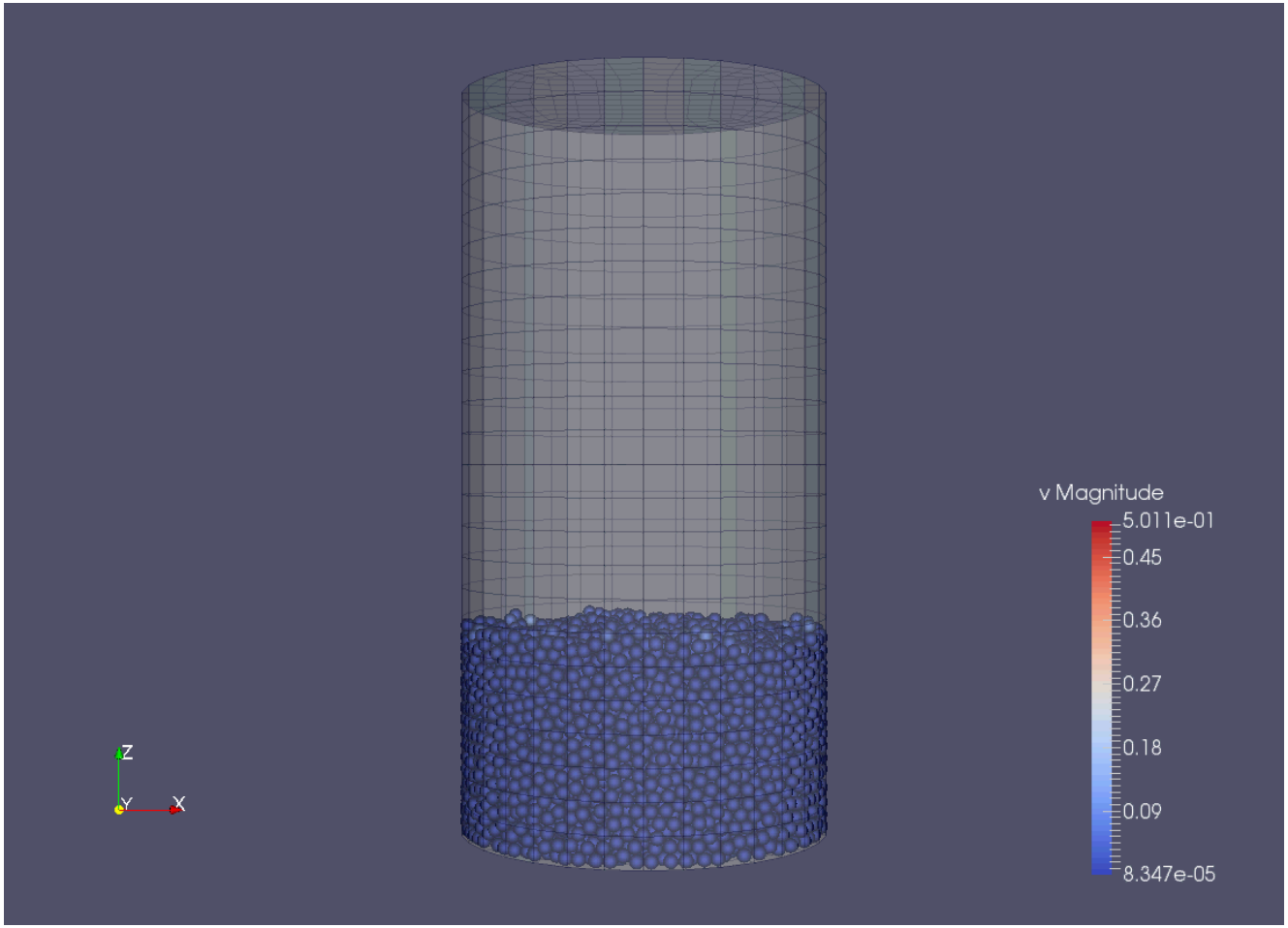
- **CFDEM coupling is an open source CFD-DEM engine. It provides the possibility to couple the DEM engine LIGGGHTS® to a CFD framework. It features:**
  - implementations for resolved and unresolved CFD-DEM
  - a variety of drag and interaction laws (implicit and explicit)
  - applicability to large-scale problems with MPI parallelization
  - its modular approach allows users to easily implement new models

CFD-DEM Model Configuration

- **Need to setup the CFD (OpenFOAM) case and DEM (LIGGGHTS) case separately.**
- **Pay attention to the coupling parameters between the two methods**
  - For “unresolved” CFD-DEM, the fluid cell size must be larger than the particle size
  - The number of DEM cycles per FVM cycle can be relatively larger if the simulation result is still reasonable.



A Fluidized Bed Example



PBS Script for CFDEM Example

```
#!/bin/bash
#PBS -q workq
#PBS -A loni_loniadmin1
#PBS -l nodes=1:ppn=20
#PBS -l walltime=01:00:00
#PBS -j oe
#PBS -N CFDEM
casePath=$PBS_O_WORKDIR
# check if mesh was built
if [ -d "$casePath/CFD/constant/polyMesh/boundary" ]; then
    echo "mesh was built before - using old mesh"
else
    echo "mesh needs to be built"
    cd $casePath/CFD
    blockMesh
fi
#- run parallel CFD-DEM in new terminal
bash $casePath/parCFDDEMrun.sh
```

LIGGGHTS Script

```
# Pour granular particles into cylinder, then induce flow
atom_style          granular
atom_modify         map array
communicate         single vel yes
boundary  m m m
newton              off
units               si
processors          2 2 5

#read the restart file
read_restart        ../DEM/liggghts.restart
#do not do this here, the simulation box is in the restart file!
#region             reg block -0.015 0.015 -0.015 0.015 -0.001 0.0554 units box
#create_box         1 reg
neighbor  0.0005 bin
neigh_modify        delay 0
#Material properties required for new pair styles

fix                 m1 all property/global youngsModulus peratomtype 5.e6
fix                 m2 all property/global poissonsRatio peratomtype 0.45
```

Some CFD Settings (OpenFOAM)

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
application     pisoFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         0.6;
deltaT          0.0001;
writeControl    adjustableRunTime;
writeInterval   0.01;
...
```

Phase Coupling Parameters

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2                                   |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
...
// sub-models & settings
modelType "A"; // A or B
couplingInterval 100;
voidFractionModel divided;//centre;//
locateModel engine;//turboEngineM2M;//
meshMotionModel noMeshMotion;
regionModel allRegion;
IOModel basicIO;
probeModel off;
dataExchangeModel twoWayMPI;//twoWayM2M;//twoWayFiles;//oneWayVTK;//
averagingModel dense;//dilute;//
...
```

Other Popular Engineering Packages

- **Commercial**
  - ABAQUS
  - ADINA
  - LS-Dyna
  - CD-Adaptco
- **OpenSource**
  - Deal II
  - OpenLBM
  - Clawpack

Some Tips for Running on HPC

- **Always start from a small problem then gradually increase the scale**
- **Use interactive session to debug your problem input and script**
- **Monitor your node/memory/CPU usage during the job run**

Summary

- **Types of engineering packages available for you to solve the PDEs that requires HPC**
- **Common scenarios for solution of engineering problems using a package**
- **Examples of running representative commercial and open source packages**

Next Week Training

- **Version Control with GIT, Nov. 11**
  - Version Control is the management of changes to documents, computer programs, large web sites, and other collections of information. This tutorial gives an introduction to the Git version control software and will cover the following topics:
    - setting up a repository
    - workflows: checkout, update and making changes to the repository
    - committing and analyzing changes to the repository, resolving conflicts

- **Weekly trainings during regular semester**
  - Wednesdays “9:00am-11:00am” session, Frey 307 CSC

- **Programming/Parallel Programming workshops**
  - Usually in summer

- **Keep an eye on our webpage: www.hpc.lsu.edu**