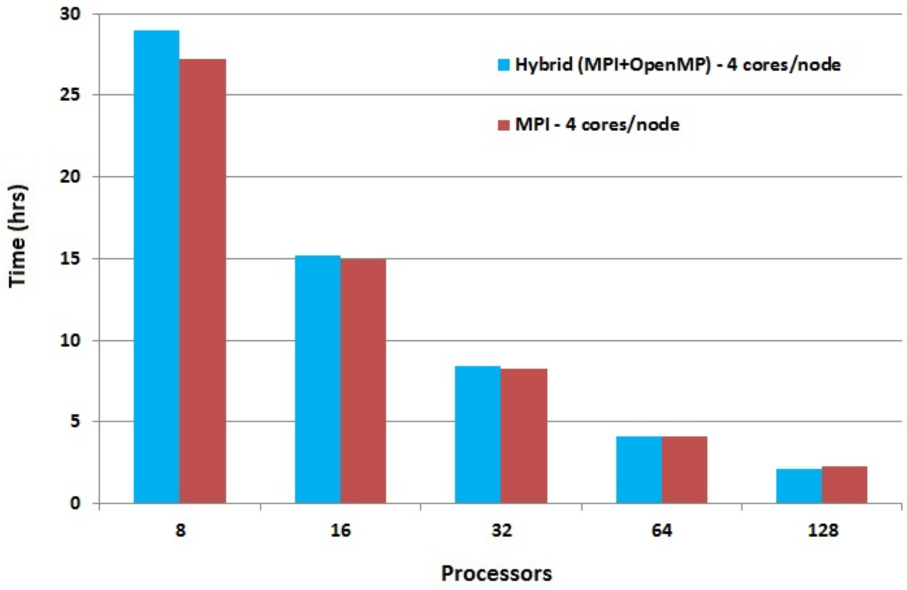


Development of OpenMP Parallelization for MFIX-DEM

Dr. Handan Liu
Research Scientist
Virginia Tech

Dr. Danesh Tafti
W.S. Cross Professor
Virginia Tech

VirginiaTech
Invent the Future

U.S. DEPARTMENT OF ENERGY

NETL 2013 Workshop on Multiphase Flow Science
August 6-7, 2013, Morgantown, WV

Motivation / Objectives

- **Motivation**
  - Different parallelization strategies are being considered on modern computer architectures that can lead to large performance gains of MFIX to allow calculations on more physically realistic systems.

- **Objectives**

  MPI parallelization for MFIX-DEM has been completed in the previous work at Virginia Tech group.

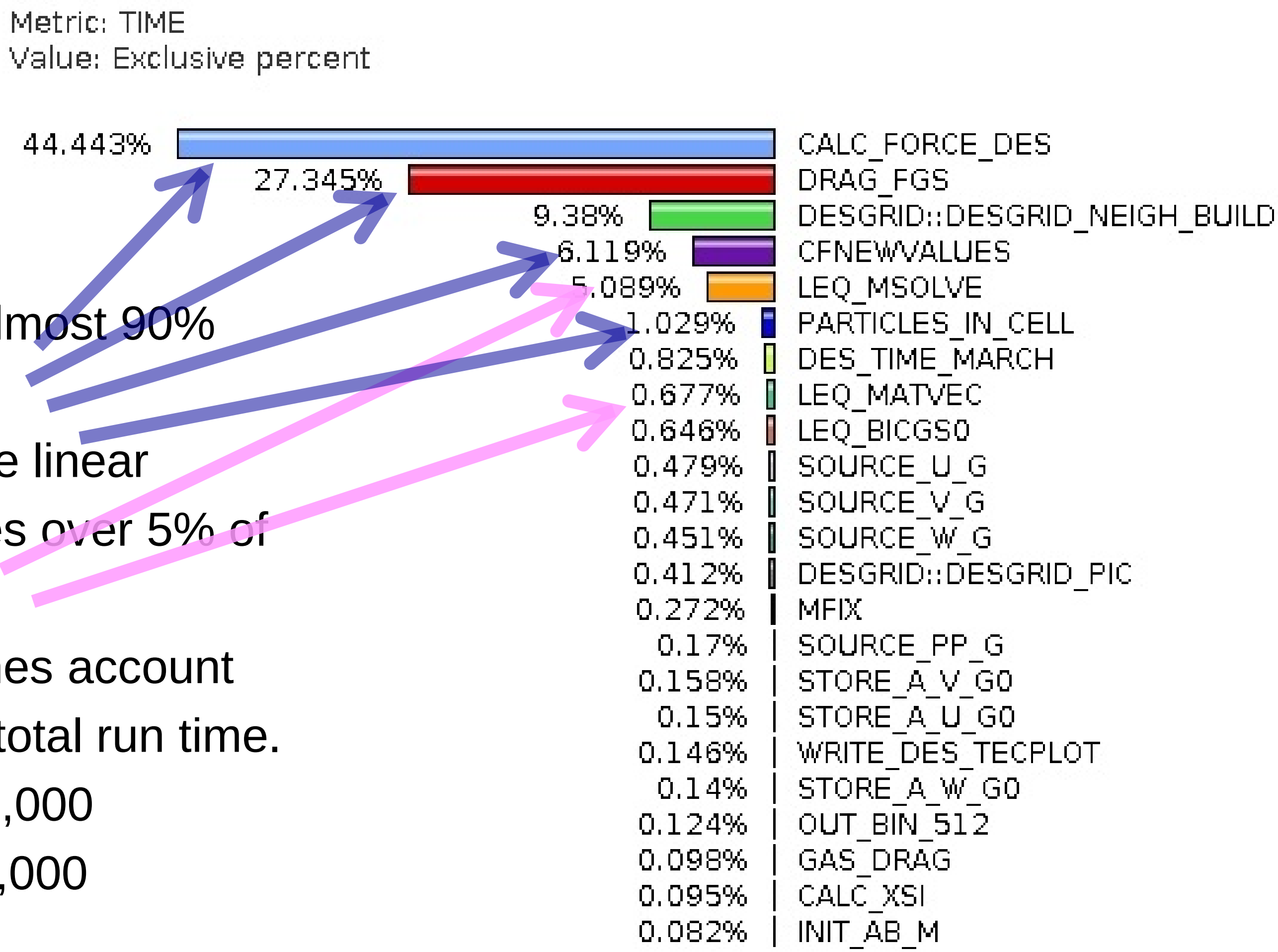
  - Enhance parallelization flexibility on multicore systems by OpenMP;
  - Improve scalability on multicore SMP cluster systems by Hybrid implementation of MPI+OpenMP;
  - Extend OpenMP instrumentation to co-processing (GPUs; Intel-MIC).

Method

- Evaluate Initial OpenMP Performance
  - TAU (Tuning and Analysis Utilities) profiling along with PDT (Program Database Toolkit)
- Identify the major (time-consuming) routines
- Analyze the data structures on loop level in these routines
- Assign variables with different attributes, e.g. private or shared, for OpenMP implementation
- Set the intermediate variables as private to replace global variables to realize OpenMP parallelization if necessary.
- Modification Principle:
  - In compliance with the existing framework of MFIX

MFIX-DEM

Initial Performance Evaluation – Exclusive Time



☞ DEM solver takes almost 90% of the total time.

☞ Preconditioner of the linear equation solver takes over 5% of the total time

☞ The major subroutines account for over 95% of the total run time.

☞ Total Particles – 80,000
Total Cells – 18,000

Major Routines for OpenMP Parallelization

- Parallelize these major routines for OpenMP implementation
  - **In DEM solver**
    - Contact force computation
      - *calc_force_des.f*
    - Drag force computation
      - *drag_fgs.f*, including *calc_des_drag_gs* and *des_drag_gs*
    - Locating the particle in fluid cell (particles in cell)
      - *particles_in_cell and comp_mean_fields_interp*
  - **In Fluid solver**
    - Linear equation solver of Bi-Conjugate Gradients Stabilized method (BiCGSTAB)
      - *leq_msolve (line iterative preconditioners)*

Typical Problems for OpenMP Parallelization in MFIX-DEM

- In MFIX-DEM, there are two main kinds of do-loops, in which care needs to be taken with OpenMP parallelization
  - One kind of do-loops is over all fluid cells; the inner do-loop is over the particles in the corresponding cell and often using the neighbor cells quantities.
    - Note: the neighbor cells maybe spread across threads for OpenMP implementation
  - Another kind of do-loops is over all particles, in which fluid cell indices are used.
    - Note: the total number of particles is spread over threads and the particles locating to a fluid cell maybe spread across threads.

Data-Sharing Attributes in OpenMP Program

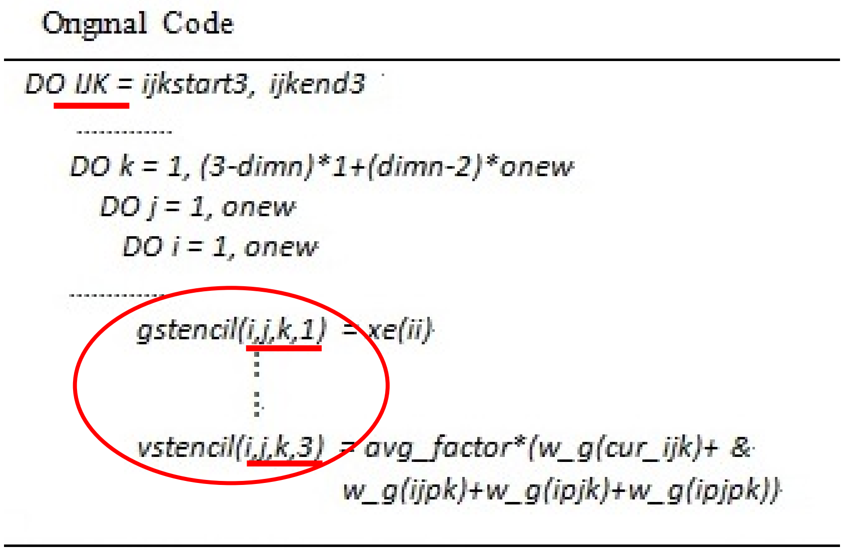
❑ *In an OpenMP program, there are two basic types*:

- SHARED
  - ✔ All threads can read and write the data simultaneously, unless protected through a specific OpenMP construct
  - ✔ All changes made are visible to all threads

- PRIVATE
  - ✔ Each thread has a copy of the data
  - ✔ No other thread can access this data
  - ✔ Changes only visible to the thread owning the data

A Typical OpenMP Example in MFIX-DEM

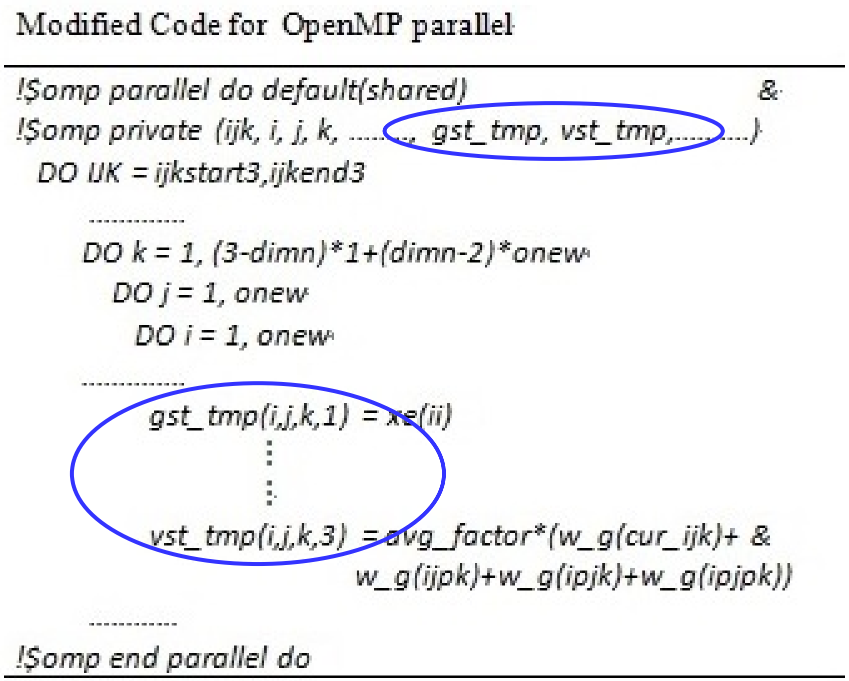
The following code fragments exist in every interpolating calculation in MFIX-DEM model



Ongnal Code

```
DO IJK = ijkstart3, ijkend3
  ...............
  DO k = 1, (3-dimn)*1+(dimn-2)*onew
    DO j = 1, onew
      DO i = 1, onew
  ...............
        gstencil(i,j,k,1)  = xe(ii)
                  :
                  :
        vstencil(i,j,k,3)  = avg_factor*(w_g(cur_ijk)+ &
                              w_g(ijpk)+w_g(ipjk)+w_g(ipjpk))
```

- The code index *IJK* loops over all fluid cells to calculate the fluid velocity interpolating at the particle location.
- For each fluid cell, the global variables *gstencil* and *vstencil* calculate the geometry and the velocity factors for interpolation.
- In the original code, because each fluid cell is visited sequentially, *gstencil* and *vstencil* will have unique values corresponding to that fluid cell before it goes to the next fluid cell.



Modified Code for OpenMP parallel

```
!$omp parallel do default(shared)                                  &
!$omp private (ijk, i, j, k, ......., gst_tmp, vst_tmp,............)
  DO IJK = ijkstart3,ijkend3
    ...............
    DO k = 1, (3-dimn)*1+(dimn-2)*onew
      DO j = 1, onew
        DO i = 1, onew
    ...............
          gst_tmp(i,j,k,1)  = xe(ii)
                    :
                    :
          vst_tmp(i,j,k,3)  = avg_factor*(w_g(cur_ijk)+ &
                               w_g(ijpk)+w_g(ipjk)+w_g(ipjpk))
    ..............
!$omp end parallel do
```

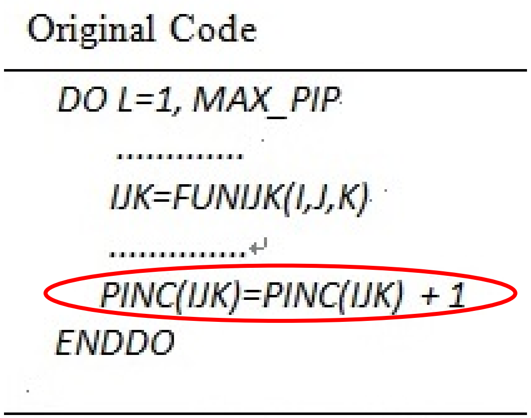
- In an OpenMP parallel do loop, *gstencil* and *vstencil* are global and shared between threads, and do not have an *IJK* dimension.
- The arrays *gstencil* and *vstencil* should be set as private to avoid a race condition by defining private arrays *gst_tmp* and *vst_tmp* for OpenMP parallel.

Another Typical Problem with OpenMP in MFIX-DEM

- Reduction Operations
  - When a variable has been declared as SHARED because all threads need to modify its value, it is necessary to ensure that only one thread at a time is writing/updating the memory location of the considered variable, otherwise unpredictable results will occur.
  - By using the clause REDUCTION it is possible to solve this problem, since only one thread at a time is allowed to update the value, ensuring that the final result will be the correct one.

Another Typical OpenMP Example in MFIX-DEM

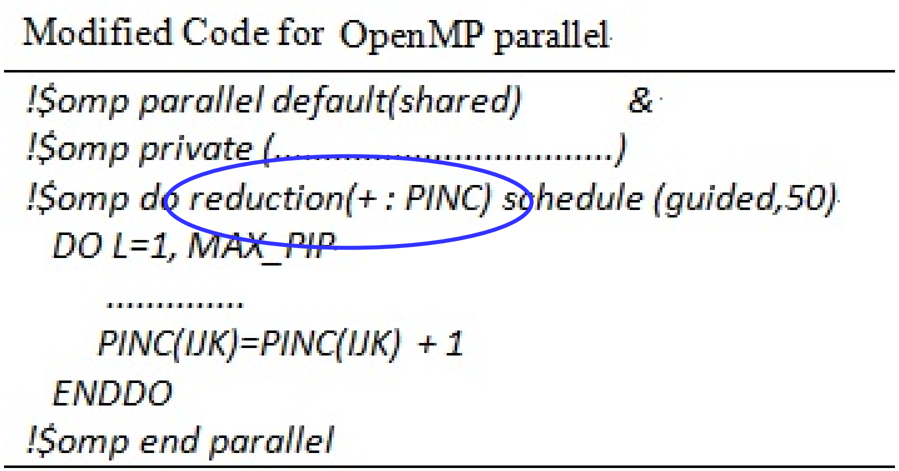
The following code fragments from the subroutine *particles_in_cell* to bin particles in fluid cells.



Original Code

```
DO L=1, MAX_PIP
     .............
     IJK=FUNIJK(I,J,K)
     ..............
    PINC(IJK)=PINC(IJK) + 1
ENDDO
```

- The code loops over all particles and bins particles in fluid cells based on particles(i, j, k) indices.
- *PINC(IJK)* is a global array to store the number of particles in a fluid cell.
- In the sequential code, each particle is visited sequentially and binned in the corresponding fluid cell to increment the value of *PINC(IJK)*.



Modified Code for OpenMP parallel

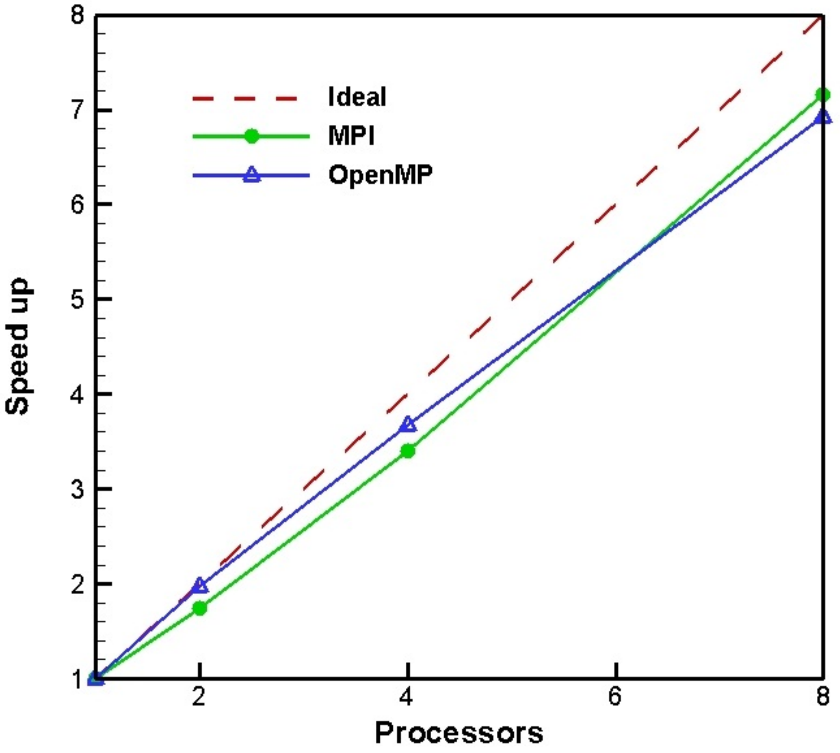
```
!$omp parallel default(shared)         &
!$omp private (................................)
!$omp do reduction(+ : PINC) schedule (guided,50)
  DO L=1, MAX_PIP

       .............
      PINC(IJK)=PINC(IJK) + 1
  ENDDO
!$omp end parallel
```

- In OpenMP implementation, the total number of particles is distributed over threads and particles belonging to a fluid cell maybe spread across threads.
- Care needs to be taken when updating shared variable *PINC*.
- With the REDUCTION clause, the OpenMP compiler generates code such that a race condition is avoided.

Scaling Analysis of OpenMP for MFIX-DEM After modifying major routines

- The 3D fluidized bed with 80,000 particles and 18,000 cells was simulated.
- A workstation:
  - ✓ Running Red Hat Enterprise Linux release 5.3
  - ✓ One node with two Intel Xeon 2.27GHz quad-core processors with a total memory of 24GB.
- Intel Compiler 13.1.
- OpenMP implementing on 2, 4 and 8 threads.

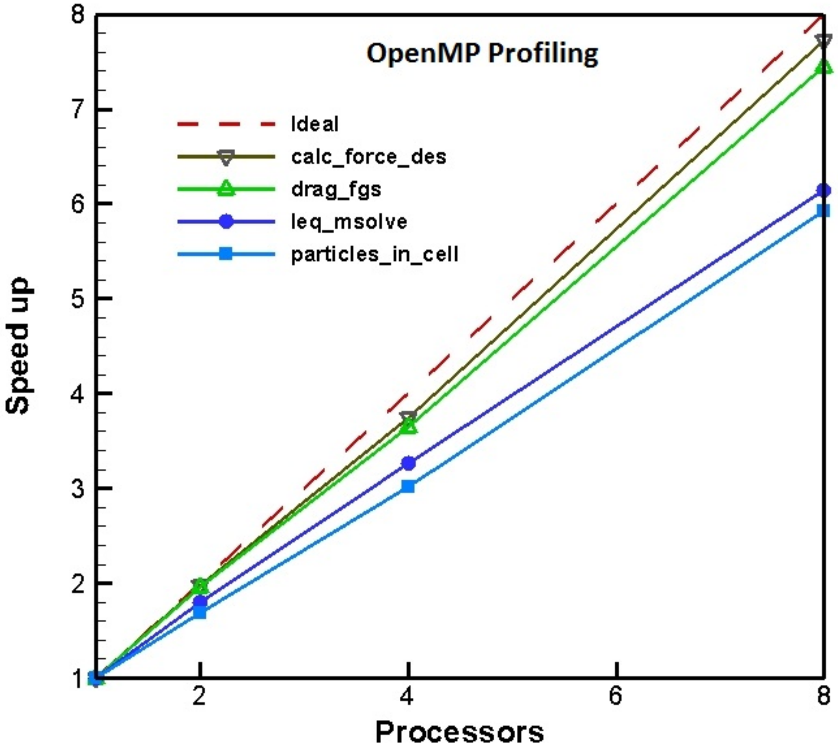


- Performance evaluation for full simulation with a speed up of 7x on 8 threads and an efficiency of 87% after modified for OpenMP parallel.

Scaling Analysis of OpenMP

– Major Routines

- Performance evaluation for major routines after modification with OpenMP directives. The speed up is as below:
  - calc_force_des.f:
    - 7.7x on 8 threads
  - drag_fgs.f
    - 7.4x on 8 threads
  - leq_msolve
    - 6.2x on 8 threads
  - particles_in_cell.f
    - 5.9x on 8 threads



☞ The 3D bubbling fluidized bed case with 80,000 particles was simulated.

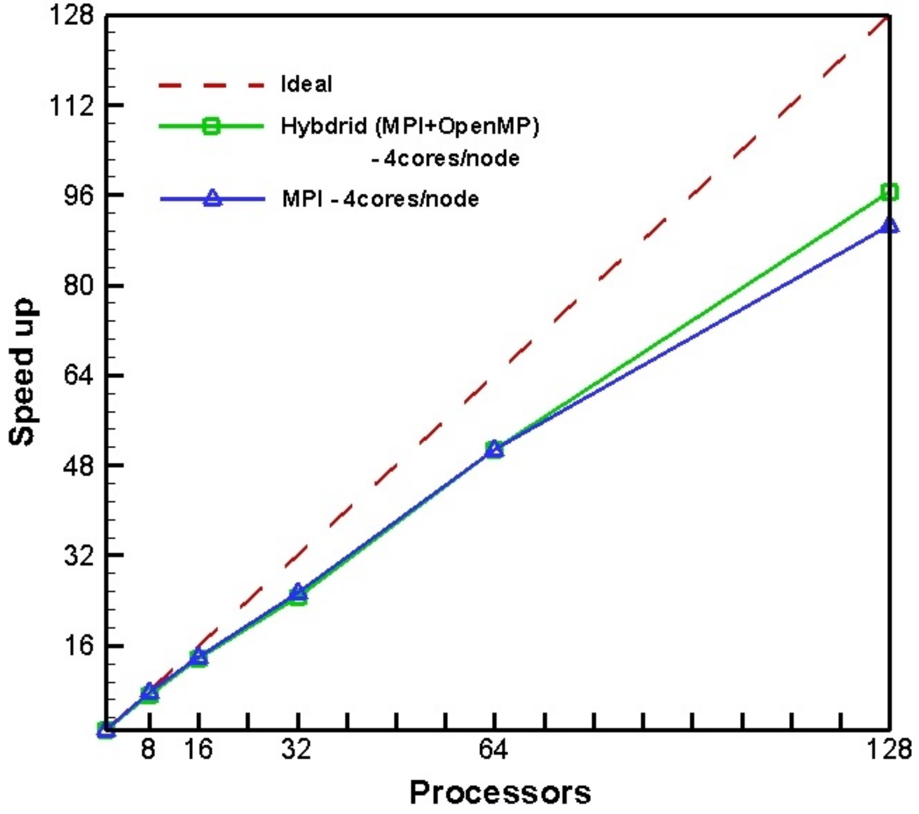
Large System – Hybrid MPI+OpenMP

- Total Particles – 1.28 million; Total cells – 409,600
- Hybrid parallelism of MPI+OpenMP implementation up to 128 processors
- SMP cluster:
  - 204 nodes Total
  - 2x Intel Xeon E5645 2.4GHz
  - 12 Cores per node
  - 24 GB Shared Memory per node
  - QDR Infiniband interconnect
- Intel Compiler 13.1
- Domain decomposition only in x and z directions for MPI
- Total physical simulation time is 0.5 seconds

Parameter	Value
Total Particles	1.28 million
Diameter	4 mm
Density	2700 kg/m^3
Coef. of restitution Particle, Wall	0.95, 0.95
Friction coefficient Particle, Wall	0.3, .03
Spring constant Particle, Wall	2400, 2400 N/m
Dimension Grid size	64×100×64 cm 64×100×64
Superficial Velocity	2.0 m/s
Time Step (Fluid, Solid)	5.0e^{-5} s, 8.6e^{-6} s
Number of processors	8,16,32,64,128

Scaling Analysis of hybrid MPI+OpenMP for large DEM system

- The hybrid calculation gives a speedup of 96x versus 89x for standalone MPI on 128 cores.
- As the number of MPI processes increases, the overhead of communicating ghost particles between MPI processes also increases.



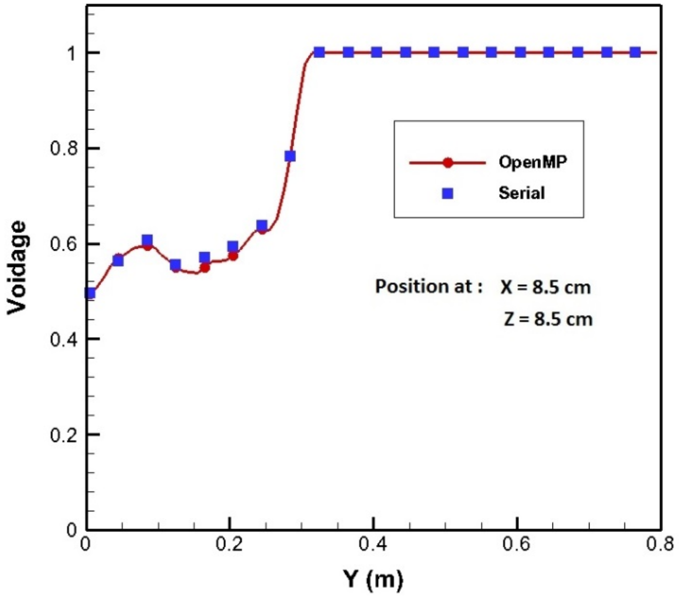
Total Particles – 1.28 million;
Total cells – 409,600

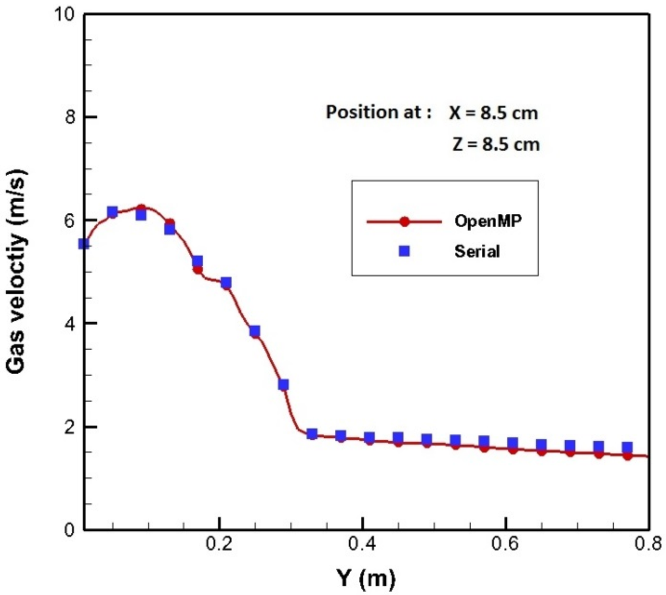
Validation

Ascertain the parallelization does not change in the physics.

- The 3D fluidized bed with 80,000 particles of 4mm diameter was simulated for this validation.
- The simulation was carried out for a total of 5 seconds.
- The time averaged profiles were obtained from 2.0-5.0 seconds.
- Results of time-averaged void fractions and gas velocity (V_g) were compared at the location x=8.5cm and z=8.5cm for serial and OpenMP (4 threads) implementation.

- *The validation shows that the parallel simulation does not alter the accuracy of the solution.*





Summary

- The parallel DEM solver for OpenMP implementation were developed for MFIX. Performance analysis was carried out to identify the time-consuming routines.
- After instrumentation with OpenMP directives, performance analysis shows an efficiency of 87% on 8 threads for a 3D MFIX-DEM bubbling fluidized bed with 80,000 particles.
- Hybrid parallelism (MPI + OpenMP) performance was evaluated for a large scale system of 1.28 million particles in a 3D bubbling fluidized bed on a large SMP cluster. The scaling analysis shows good scalability for MFIX-DEM up to 128 processors (10,000 particles/processors) with an efficiency of 75%.
- The validation of MFIX-DEM shows that the parallel simulation does not alter the accuracy of the solution.

Future Work

- **FY13 –**
  - Further scaling analysis for large scale DEM system by hybrid (MPI+OpenMP) parallel programming on large processor counts
  - Investigate MPPIC model with OpenMP directives
- **FW – Extend to co-processing architectures (GPUs, Intel-MIC)**

Publications

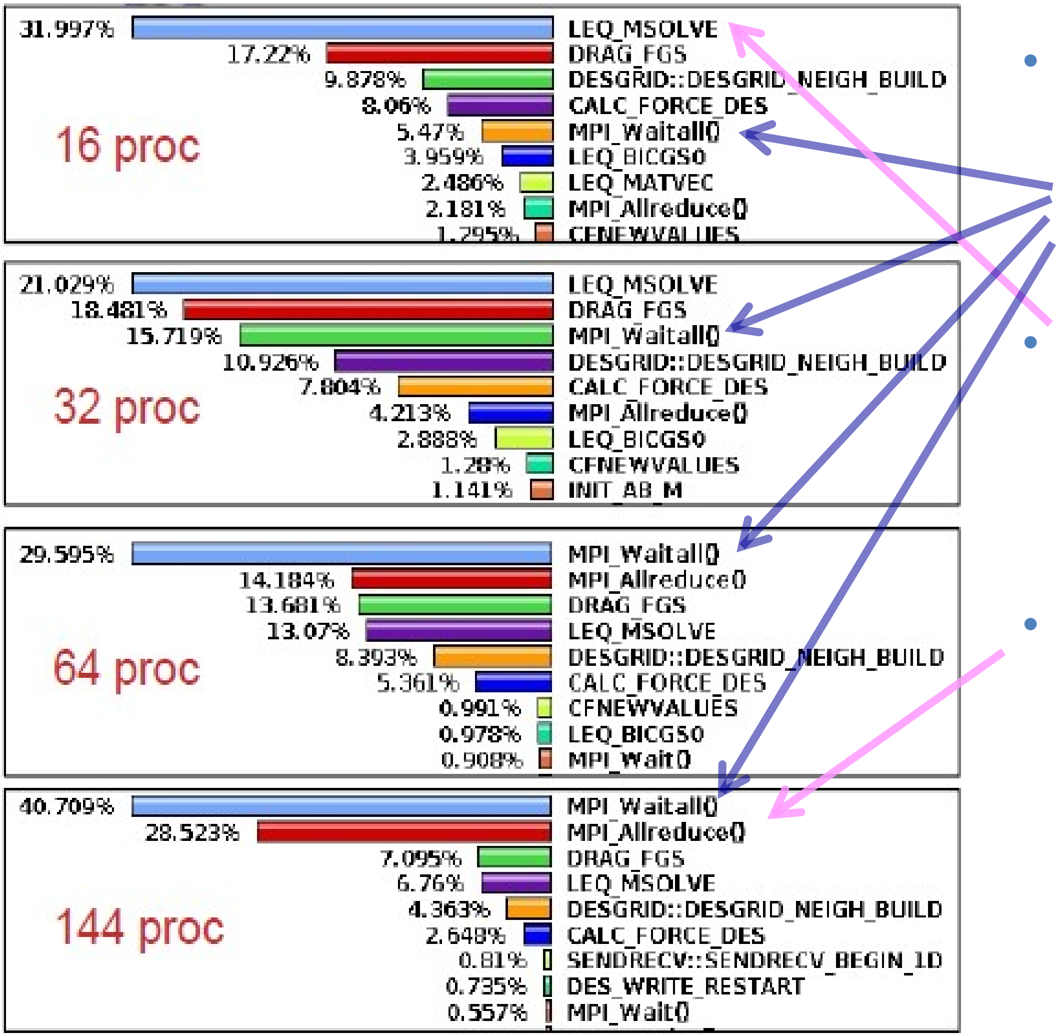
- [1] Handan Liu, Danesh Tafti and Tingwen Li. Hybrid Parallelism in MFIX CFD-DEM using OpenMP. Powder Technology. Under review.
- [2] Handan Liu, Danesh Tafti. Summary Report for MFIX Acceleration. Dec. 2012. Project Title: MFIX acceleration (number: 683.232.001).
- [3] Handan Liu, Danesh Tafti. Summary Report for MFIX Acceleration on New Code. March 2013. Project Title: MFIX acceleration (number: 683.232.001).
- [4] Handan Liu, Danesh Tafti. Summary Report for MFIX Acceleration on MPPIC. August 2013. Project Title: MFIX acceleration (number: 683.232.001).
- [5] Pradeep G., Danesh Tafti. Development of parallel DEM for the open source code MFIX. Powder Technology, 235 (2013) 33-41.
- [6] Amit Amritkar, Danesh Tafti, Rui Liu, Rick Kufrin, Barbara Chapman. OpenMP parallelism for fluid and fluid-particulate systems. Parallel Computing, 38 (2012) 501-517.

Thank You !

Questions?

Supplement

Scaling Analysis – Exclusive Time



- As number of processors increases time required for MPI communication increases
- For 16 processors hydrodynamic solver takes most of the time
- For 144 processors interface exchange takes 40% and MPI collective communication takes 28% of total time

Pradeep Gopalakrishnan, Danesh Tafti, Parallelization of Discrete Element Method, ORD 2.672.232.001.000 Merit Review, April 26, 2011