**Abstract**

Fluid flow through porous materials is at the center of many engineering and scientific applications such as oil production and filtration processes. The significant progress in high-performance computing in recent decades has immensely aided the research in these fields, especially by providing a rich level of details about the transport phenomena occurring at pore-scale level. In this work, the goal is to use the versatile open-source multiphysics package OpenFOAM to solve the flow in porous media problem. This CFD toolbox – capable of parallelization – provides the user with a variety of solvers for different physical problems. Moreover, all the pre-processing steps involved in solving this difficult problem (mesh generation, domain decomposition) can be performed in OpenFOAM. The problem of the laminar flow of an incompressible fluid through a random, polydispersed granular pack is studied. As in any other numerical study, it is necessary to understand the scaling possibilities and the bottlenecks of the codes. The set-up of the problem along with the results of profiling the solver with the software IPM (Integrated Performance Monitoring) are provided for a variety of cases.

**Methods & Hardware**

The porous medium used in this work is consisted of 1,000 spherical particles of varying diameters \( \approx 24-114 \mu \text{m} \) with a porosity of 37%. The fluid flow in this problem can be modeled using the Navier-Stokes equations of motion. The boundary conditions in this problem are:

1. Imposed pressure gradient along one of the cartesian coordinates (z-axis).
2. No-slip boundary condition at any solid (grain) surface.
3. The velocity at the inlet surface is computed using the known pressure field at that patch from the flux at a normal direction to the inlet faces.

The mesh used in this study is generated using the SnappyHexMesh utility of OpenFOAM. This utility is not designed to create volume meshes of complicated, three-dimensional void spaces of varying sizes which are common in any porous material. Therefore, certain considerations need to be taken regarding the input values for a few of the parameters of this utility (surface refinement, feature angle, and snap controls). Figure 1 shows the result of the generated mesh.

The problem is solved on LSU’s SuperMike-II cluster using the workq nodes, each of which consists of:

- Two 2.6 GHz 8-Core Sandy Bridge Xeon 64-bit Processors
- 32GB 1666MHz Ram
- 500GB HD
- 40 Gigabit/sec Infiniband network interface
- 1 Gigabit Ethernet network interface
- Red Hat Enterprise Linux 6

**Profiler Results and Conclusions**

- As more and more processors are added, efficiency is lost to startup (which includes loading the mesh) and the region of the code which was not profiled (likely communication between processes in between profiled sections).
- There was a large growth in the percentage use of MPI_Probe and MPI_Allreduce as the number of processors increased.
- This bottleneck usage of MPI_Probe is likely a result of inconsistent size distributions of data being transmitted between processes. A method to keep track of the size of data (or amount of elements being sent and received) would greatly improve the speed and performance of the icoFoam solver.
- It is known that OpenFOAM is not optimized to avoid cache misses\(^1\). Improvements in data storage and cache optimization may be able to reduce the MPI communication overhead.

**References**


**Future Research**

Unfortunately due to outside constraints, the project was unable to proceed and do runs on Intel Xeon Phi Coprocessors. In the future, work must be done to compile OpenFOAM on such an architecture and further studied.

**Grant**

This material is based upon work supported by the National Science Foundation under award OCI-1263236 with support from the Center for Computation & Technology at Louisiana State University.