Performance Optimization of OpenFOAM on the new Intel® Xeon Phi™ Processor

Ravi Ojha[1], Prasad Pawar[1], Nishant Agrawal[1], Ambuj Pandey[1]
Paul Edwards[2]

[1] *Tata Consultancy Services Ltd, Gateway Park, MIDC Road No#13, Andheri East, Mumbai, Maharashtra 400093, India {rav.ojha, prasad.pawar,* nishant.agrawal*, ambuj.pandey,}@tcs.com*

[2] *Intel Corporation (UK) Ltd., Pipers Way, Swindon, SN3 1RJ, paul.m.edwards@intel.com*

OpenFOAM* is a well-known and popular software package for solving partial differential equations (PDE) and is used by industry, researchers and academia to solve variety of computational fluid dynamics (CFD) problems. It comes with several different kinds of solvers for different PDEs, but also ships with a framework to implement third-party solvers for custom PDEs. OpenFOAM uses a finite volume method (FVM) and employs the Message Passing Interface (MPI) for communication; as of today it does not yet support multi-threading.

Recent advances in parallel computing have enabled solving time-consuming problems in a reasonable time frame or to tackle larger problems, that is, more detailed analysis of existing structures or simulation of more complex and larger structures. Technologies include multicore CPUs with few but powerful individual cores that strive to balance single-threaded and multi-threaded performance, and GPUs that have a higher number of cores although less powerful and designed to work in lock-step with all other cores executing the same instruction on different elements of the data sets.

Intel recently launched Knights Landing (KNL) based on their Many Integrated Core (MIC) Xeon Phi™ processor. Compared to the traditional multi-core systems, the cores of KNL are clocked at a lower frequency. Unlike the earlier versions where coprocessors were used as PCIe accelerators, KNL is now available as a bootable CPU. The latest hardware has an on-die Omni Path interconnect for high bandwidth. The hardware also offers high bandwidth stacked die memory of 16GB with bandwidth of over 400GBps.

This paper describes the work done to optimize the performance of OpenFOAM on the above mentioned hardware by writing Advanced Vector Extensions 512 (AVX512) intrinsics. Analyzing the vectorization report of our top hotspot function GaussSeidel revealed that the compiler was generating Peel and Remainder loops because the trip count of the loop is unknown. Moreover analyzing the assembly code in Intel® VTune™ showed that a few unnecessary instructions were getting generated as well. These gave us the motivation to write our own AVX512 intrinsics for doing the vectorization and achieving a 13% overall runtime of our benchmark simulation.

Besides intrinsics, we present other optimization work on OpenFOAM which is now accepted in the OpenFOAM main branch where we reduce the application startup time for large numbers of cores and we reduced the peak memory usage by 15% in our benchmarks.