Contribution ID: **14** Type: **not specified**

Vector Parallelism for Kalman-Filter-Based Particle Tracking on Multi- and Many-Core Processors

Thursday, 13 July 2017 08:30 (2 hours)

All modern CPUs boost their performance through vector processing units (VPUs). Typically this gain is achieved not by the programmer, but by the compiler through automatic vectorization of simple loops in the source code. Compilers generate SIMD instructions that operate on multiple numbers simultaneously by loading them together into extra-wide registers. Intel's latest processors feature a plethora of vector registers, as well as 1 or 2 VPUs per core that operate on 16 floats or 8 doubles in every cycle. Vectorization is an important component of parallel performance on CPUs, and to maximize performance, it is vital to consider how well one's code is being vectorized by the compiler.

In the first part of our presentation, we look at simple code examples that illustrate how vectorization works and the crucial role of memory bandwidth in limiting the vector processing rate. What does it really take to reach the processor's nominal peak of floating-point performance? What can we learn from things like roofline analysis and compiler optimization reports?

In the second part, we consider how a physics application may be restructured to take better advantage of vectorization. In particular, we focus on the Matriplex concept that is used to implement parallel Kalman filtering in our group's particle tracking R&D project. Drastic changes to data structures and loops were required to help the compiler find the SIMD opportunities in the algorithm. In certain places, vector operations were even enforced through calls to intrinsic functions. We examine a suite of test codes that helped to isolate the performance impact of the Matriplex class on the basic Kalman filter operations.

Presenters: LANTZ, Steven R (Cornell University (US)); TADEL, Matevz (Univ. of California San Diego (US))