Vectorization of Chapel Code

Elliot Ronaghan, Cray Inc. (presenting author)

Vectorization is crucial for achieving peak performance on modern commodity and HPC systems. Vectorization will become even more important for processors using AVX3 or NEON instructions sets including Intel's Xeon Phi Coprocessors that have extremely wide 512 bit registers for vector instructions.

Unfortunately, modern programming languages such as Go or Rust provide very little in the way of vectorization. Traditional HPC languages such as C and C++ provide a better story with OpenMP 4.0 SIMD constructs, vendor specific intrinsics, and auto-vectorization. However, C and C++ compiler are often thwarted from auto-vectorizing code because of pointer aliasing. Achieving peak performance often requires users to write explicitly vectorized code that might not be portable across compilers or architectures. Additionally, depending on the code and architecture, vectorization might not always increase performance so OpenMP 4.0 like approaches might not be an ideal solution.

With it's parallel and array features as well as lack of pointers, Chapel is uniquely suited to allow users to write code for HPC and commodity hardware that should vectorize well without putting as much burden on users as current languages do.

This talk will present recent work to enable better vectorization of Chapel code. This includes generating clean C style for loops, eliminating range construction for anonymous ranges, and conveying information about loops that are known to be order independent by the Chapel compiler to the backend compiler. This will also explore planned future work and longer term ideas that are still in the works with the hope of starting community discussion on vectorization techniques and potential benchmarks.