Abstract

This work describes the challenges presented by porting parts of the Gysela code to the Intel Xeon Phi coprocessor, as well as techniques used for optimization, vectorization and tuning that can be applied to other applications. Several interpolation kernels useful for the Gysela application are analyzed and the performance are shown. Some memory-bound and compute-bound kernels are accelerated by a factor 2 on the Phi device compared to Sandy architecture. Nevertheless, it is hard, if not impossible, to reach a large fraction of the peek performance on the Phi device, especially for real-life applications as Gysela. A collateral benefit of this optimization and tuning work is that the execution time of Gysela (using 4D advections) has decreased on a standard architecture such as Intel Sandy Bridge.

Reference

http://arxiv.org/abs/1503.04645