Book Review:

Intel Xeon Phi Coprocessor High Performance Programming

James Jeffers, James Reinders
Morgan Kaufmann, 2013
ISBN-13: 978-0124104143

James Reinders (Chief Evangelist of Intel® Software at Intel) and Jim Jeffers (Principal Engineer at Intel) have gathered in this book the working knowledge of the people involved to prototype and productize the Intel® Xeon Phi coprocessor. This include the experience of Intel's Product, Field, Application and Technical Consulting Engineers, as well as key partners involved on product pre-release work.

The book offers a hands-on and practical guide, presenting best-known-methods for High Performance Computing programming with the Intel Xeon Phi coprocessor. The book's contents can be split in four main sections: the usual introduction, an interesting optimization guide, plus product architecture details and, before concluding, a review of available tools and libraries.

The initial section introduces Intel Xeon Phi through analogies to a racing car. *Chapter 1 – Introduction* details when, why and how to use Intel Xeon Phi coprocessors. This chapter starts applying an analogy of how sports cars are designed and how they behave in a variety of situations. One key point of this introduction is that learning how to optimize code to expose parallelism will be useful to any other processor, not only Intel Xeon Phi. *Chapter 2 - High Performance Closed Track Test Drive* takes the sports cares analogy to a test with a sample hello world code with the usual SAXPY computation. Through real command line examples takes the reader from the C compiler invocation, passing for the checking of the vectorization results, up to the actual execution. Surprisingly, all of this happening in a Linux image running inside the coprocessor card. Discussion continues showing how to use multithreading computation using OpenMP extensions. How to offload work directly from a host-system is also shown, double checking that gathered performance is about the same. *Chapter 3 - A Friendly Country Road Race* moves the sports car out of the synthetic and controlled environment to solve a 9-point stencil algorithm. On this case even running over multiple coprocessors using MPI capabilities. Then low-level tuning start with access to memory taking into account memory alignment, streaming stores and bigger memory pages. *Chapter 4 - Driving Around Town* uses a short but still representative example on how to vectorize a loop by exposing parallelism properly, focusing on data locality and data tiling to favor cache reuse.

The second section goes into detail on parallelization techniques. A good aspect to take into account is these optimization techniques might end up favoring any multicore system. *Chapter 5 - Lots of Data (Vectors)* gives helpful vectorization insights, proposing a simple systematic procedure that can be iterated over the code. For instance, manual loop unrolling is disregarded here as it is more future-proof to guide the compiler instead of manual re-tuning when architecture evolves. *Chapter 6 - Lots of Tasks (not Threads)* takes the parallelism abstraction up level, up to computation threads. Here it is pointed out that thread creation should happen at the microprocessor, thread launch nesting is discussed and shown with language extensions like OpenMP and FORTRAN arrays, plus abstraction libraries like Intel® Threading Building Blocks, Cilk and the well-known Intel® Math Kernel library.

The third section shows product specific details, showing how to offload work and exciting internals on the available hardware and computation units. *Chapter 7 – Offload* discusses available offload models, the code can be executed natively, offloaded manually or even automatically. This enables asynchronous computation when offload is properly scheduled. *Chapter 8 - Coprocessor Architecture* touches bare metal architecture, providing details about cache organization, vector processing units, direct memory access and available power management interfaces. *Chapter 9 - Coprocessor System Software* details low level libraries used to exchange information between the host system and the coprocessor. These libraries abstract PCI mapped memory as a regular networking device, allowing usual IPC mechanisms to work against the coprocessor. Memory allocation at operating system kernel level is discussed as it provides yet another optimization possibility.

Chapter 10 - Linux on the Coprocessor provides Linux specifics of the image running inside the processor, this fact enables easy extension of the available software, as Open Source tools and libraries might be compiled natively and included inside the Linux image.

A fourth section explain how to use the multiple libraries and tools made available by Intel to increment productivity. Not only for compiling code, but also graphical tools to review profiling information of code running on the coprocessor. This tools were built specifically for the product, with negligible overhead thanks to the use of special purpose hardware counters. *Chapter 11 - Math Library* and *Chapter 12 – MPI* shows how the well-known software now supports Intel Xeon Phi, offering specific optimizations and abstractions to de-facto interfaces like BLAS and FFT. *Chapter 13 - Profiling and Timing* makes a useful summary of profiling information such as expected CPI, monitored events and efficiency metrics to look at while reviewing performance.

In a nutshell, this book covers parallel programming with Intel Xeon Phi. The book is reader friendly, both for the novice and expert, as it includes tons of analogies and examples with real-world code. Luckily, the end of the book specifies that a Volume 2 is coming with even more low level details to exploit every available computing capability left. Lots of pointers to relevant publications are included on each chapter during the discussion of presented ideas, it is up to the reader to follow them and get in-deep details.

Disclaimer: the author of this review currently works for Intel Corp. as a Senior Software Engineer. He personally collaborated with draft content of the book sections about Intel Cluster Ready and Intel Cluster Checker, projects on which contributed for more than 5 years.

Andres More
Intel Corporation (andres.more@intel.com)
Instituto Universitario Aeronáutico (amore@iua.edu.ar)