Title:
CPUs, GPUs, FPGAs: A Tutorial on Heterogeneity and Managing Accelerators with Intel Threading Building Blocks

Presenters
Michael Voss, michaelj.voss@intel.com

Pablo Reble, pablo.reble@intel.com

Rafael Asenjo, asenjo@uma.es

Abstract

Heterogeneous computing is seen as a path forward to deliver the energy and performance improvements needed over the next decade. Heterogeneous systems augment CPUs (Central Processing Units) with GPUs (Graphics Processing Units), FPGAs (Field Programmable Gate Arrays) and other resources that are able to accelerate important computations while consuming less energy. There are also heterogeneous architectures on-chip, like the processors developed for mobile devices (laptops, tablets and smartphones) comprised of multiple cores and a GPU. More recently, some architectures have also paired multicores along with an FPGA on the same die.

This tutorial first covers important hardware aspects of these heterogeneous architectures. We briefly discuss the underlying architecture of some heterogeneous chips composed of multicores + GPU and multicores + FPGA, delving into the differences between both kind of accelerators and how we can measure the energy they consume. We then move to the software side of the tutorial, where different heterogeneous programming models will be introduced, paying more attention to those that are aimed at exploiting several devices at the same time (CPU + GPU or CPU + FPGA). Again, the different optimization techniques and the levels of parallelism that are suitable for the GPU and for the FPGA will be identified.

After providing a survey of available programming models, we will focus on Intel® Threading Building Blocks (Intel® TBB). TBB is a widely used, portable C++ template library for parallel programming, that provides generic parallel algorithms, concurrent containers, a work-stealing task scheduler, a data flow programming abstraction, low-level primitives for synchronization and thread local storage and a scalable memory allocator. The generic algorithms in TBB capture many of the common design patterns used in parallel programming. While Intel TBB was first introduced in 2006 as a shared-memory parallel programming library, it has recently been extended to support heterogeneous programming. These new extensions allow developers to more easily coordinate the use of accelerators such as integrated and discrete GPUs, attached devices such as Intel® Xeon Phi co-processors, and FPGAs in to their parallel C++ applications. We will briefly cover the basics of the TBB library before presenting deeper coverage of the new features included for heterogeneous programming. Attendees can take part in hands-on exercises to create a small example and evolve it from a host-only shared-memory

implementation to a heterogeneous implementation that runs on both the host and an accelerator.

Finally, we discuss some experimental parallel patterns implemented on top of TBB. These heterogeneous implementations of the pipeline and parallel_for templates automatically distribute the workload between the multicore and the accelerator, balancing load and considering energy consumption in the scheduling policies. We evaluate the performance and energy efficiency of the different approaches for several heterogeneous processors: Intel® Xeon™, Samsung Exynos 5 Octa, Xilinx® Zynq and Altera Cyclone V.

Goals

By the end of the tutorial, attendees will be familiar with the important architectural features of commonly available accelerators, will have an understanding of how to measure performance and energy on these systems, and will have a sense of what optimizations and types of parallelism are suitable for these devices. Attendees will have also been introduced to Intel TBB, learned about its heterogeneous programming features, and will have been given the opportunity to build and execute a hybrid application. Finally, attendees will learn about ongoing research in developing hybrid schedulers for parallel patterns such as pipeline and parallel_for.

Prerequisite Knowledge

Attendees should be comfortable programming in C++ using modern features such as templates and lambda expressions. Attendees should also have an understanding of basic parallel programming concepts such as threads and locks. No previous experience with Intel® Threading Building Blocks, GPUs or FPGAs is required.

Outline:

Part 1: Motivation and background
- An introduction to heterogeneous architectures
- Important features of different accelerators such as GPUs and FPGAs
- How to measure performance and energy
- A survey of heterogeneous programming models
- How to determine if a computation is suitable for an accelerator

Part 2: Intel TBB and its heterogeneous features

- Overview of the philosophy and shared-memory features of the library
- Deep-dive in to the flow graph and its heterogeneous features

- Hands-On Exercises
  - “Hello TBB” verifying that the environment is set up correctly
  - Implement small example as a shared-memory flow graph

Part 3: Using heterogeneous flow graph nodes to coordinate accelerators

- A deep dive in to the nodes to be used in examples
- Using async_node to do asynchronous communication
- Using streaming_node and the OpenCL factory to access integrated graphics and FPGAs
- Hands-On Exercises
  - Adding async_node to the example to overlap an asynchronous computation
  - Adding streaming_node to use an OpenCL-compatible device
  - Targeting an FPGA

Part 4: Heterogeneous templates on top of TBB

- Overview of hybrid scheduling, goals and challenges
- Description of the hybrid pipeline and parallel_for implementations
- Experimental evaluation

About the presenters

Michael Voss is a software engineer in the Developer Products Division (DPD) of the Software and Services Group (SSG) at Intel, and has been a member of the Intel® Threading Building Blocks (Intel® TBB) team since 2006. He is the architect of the Intel TBB flow graph API and is one of the developers of Flow Graph Analyzer, a tool for creating and analyzing the performance of parallel, graph-based applications. He has written over 40 published papers and articles on parallel programming topics. Prior to joining Intel, he was an assistant professor in the Edward S. Rogers Sr. Department of Electrical and Computer Engineering at the University of Toronto. He received his PhD in Electrical Engineering from Purdue University in 2001. His interests include shared memory parallel programming, heterogeneous parallel programming, development and analysis of graph-based applications, compilers and runtime optimization.

Pablo Reble is a software engineer in the Developer Products Division (DPD) of the Software and Services Group (SSG) at Intel, working on Intel® Threading Building Blocks (Intel® TBB) and Flow Graph Analyzer. He received his PhD in Computer Engineering from RWTH Aachen and has more than 7 years of experience in teaching parallel programming on undergraduate and graduate level. He has authored over 10 peer reviewed papers related to system software for many-core architectures. His interests include parallel computer architectures, parallel programming as well as runtime development and optimization.

Rafael Asenjo received his B.S. and M.S. degrees in Telecommunications Engineering in 1993 and his Ph.D. degree in 1997, both from the University of Malaga, Spain. He has been an Associate Professor at the Dept. of Computer Architecture, Univ. Malaga, Spain, since 2001, where he leads a research group working on "productivity" in the context of high performance computing. He collaborated on the IBM XL-UPC compiler and on the Cray's Chapel runtime. In the last five years, he has focused on productively exploiting heterogeneous chips. In 2013 and 2014 he visited UIUC to work on CPU+GPU chips. In 2015 and 2016 he also started to work on CPU+FPGA chips while visiting U. of Bristol. He served as General Chair of ACM PPoPP'16 and has also served as Program Committee member for PACT'17, EuroPar'17 and SC'15. His research interests include heterogeneous programming models and architectures, parallelization of irregular codes, energy consumption and parallelizing compilers.

Links to information about Intel® Threading Building Blocks
http://www.threadingbuildingblocks.org

The Special Issue of Parallel Universe Magazine, "Intel® Threading Building Blocks Celebrates 10 Years!" https://goparallel.sourceforge.net/wp-content/uploads/2016/06/ParallelUniverseMagazine_Special_Edition_v2.compressed.pdf

Vasanth Tovinkere and Michael Voss, "Flow Graph Designer: A Tool for Designing and Analyzing Intel® Threading Building Blocks Flow Graphs", 2014 43nd International Conference on Parallel Processing Workshops (ICCPW), p. 149-158, 2014 http://doi.org/10.1109/ICPPW.2014.31

Michael Voss, Vasanth Tovinkere and Pablo Reble "CPUs, GPUs, FPGAs: Managing the alphabet soup with Intel Threading Building Blocks" Tutorial at PPoPP 2017, Material available online: https://github.com/01org/tbb/tree/tbb_2017_PPoPP17