Expressing Heterogeneous Parallelism in C++ with Intel Threading Building Blocks

A full-day tutorial proposal for SC17

Tutorial Instructors
[James Reinders, Michael J. Voss, Pablo Reble, Rafael Asenjo]

1. Abstract

Due to energy constraints, high performance computing platforms are becoming increasingly heterogeneous, achieving greater performance per watt through the use of hardware that is tuned to specific computational kernels or application domains. It can be challenging for developers to match computations to accelerators, choose models for targeting those accelerators, and then coordinate the use of those accelerators in the context of their larger applications.

This tutorial starts with a survey of heterogeneous architectures and programming models, and discusses how to determine if a computation is suitable for a particular accelerator. Next, Intel® Threading Building Blocks (Intel® TBB), a widely used, portable C++ template library for parallel programming is introduced. Intel TBB was first developed in 2006 as a shared-memory parallel programming library, but has since been extended to allow developers to 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.

Attendees will be given a hands-on opportunity to use TBB to create parallel implementations of a sample code. They will first build a shared-memory implementation and then a heterogeneous implementation, running their samples on a mix of CPUs and accelerators.

2. Detailed Description

This tutorial will start with a survey of heterogeneous architectures and programming models. It will introduce the Intel® Threading Building Blocks (Intel® TBB) library and discuss its relevance for HPC before presenting deeper coverage TBB's features for heterogeneous programming. Attendees will 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 CPUs and accelerators. Finally, attendees will learn about ongoing research in developing hybrid schedulers for parallel patterns such as pipeline and parallel_for.

2.1 Tutorial Goals

By the end of the tutorial, attendees will be familiar with the important architectural features of commonly available accelerators and will have a sense of what optimizations and types of parallelism are suitable for these devices. Attendees will also be introduced to Intel TBB, learn about its heterogeneous programming features, and will build and execute a hybrid application.

2.2 Relevance

Heterogeneous platforms are becoming increasingly common in HPC programming, with compute resources that include a diverse collection of integrated and discrete graphics processors, co-processors like Intel® Xeon Phi Co-processors, FPGAs and other domain-specific compute engines. Understanding the tradeoffs in using these accelerators and how to select and optimize computations for offload to these devices is an important and timely topic.

A single programming model has also not yet emerged that covers all of these devices and therefore coordinating these models and accelerators in a parallel application is challenging. C++ continues to grow in importance in HPC programming and Intel® Threading Building Blocks is a widely used C++ library. TBB offers a good platform for exploring heterogeneity and is developing as one promising solution for managing heterogeneity. By learning about TBB and its heterogeneous features, attendees will be exposed to TBB but also to the challenges in heterogeneous computing that it aims to solve.

2.3 Target Audience

Programmers in the field of High Performance Computing that want to better understand heterogeneity and to develop portable C++ applications that unleash the power of multi-core, many-core as well as heterogeneous systems.

2.4 Content level

50% beginner:	A survey of heterogeneous architectures and programming models. Loop level parallelism, Task based parallelism.
40% intermediate:	Data flow and graph parallelism. Heterogeneous features in TBB.
10% advanced:	Developing hybrid scheduling and load balancing algorithms.

2.5 Prerequisites

Attendees should have an understanding of basic parallel programming concepts such as threads and locks. Attendees should be comfortable with programming in C++. Advanced C++ features such as lambda expressions will be briefly introduce before they are used in the tutorial. No previous experience with Intel® Threading Building Blocks, GPUs or FPGAs is required.

2.6 Content

This tutorial first covers important hardware aspects of common heterogeneous architectures. We will 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.

We then will focus on Intel® Threading Building Blocks (Intel® TBB). We will briefly cover the basics of the TBB library and its relevance to HPC and heterogeneous computing. We will then present 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 accelerators.

Finally, we will 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.

2.7 Collaboration

Michael Voss and Pablo Reble are members of the engineering team that develops Intel® Threading Building Blocks. Both James Reinders and Rafael Asenjo are long-time, and continuing collaborators of the parallel software teams at Intel. The content for this tutorial is being developed collaboratively to create a unified flow and message, and will leverage existing content from previous collaborations such as our Euro-Par tutorial.

2.8 Previous Presentations

"CPUs, GPUs, FPGAs: A Tutorial on Heterogeneity and Managing Accelerators with Intel Threading Building Blocks," Michael Voss, Pablo Reble and Rafael Asenjo, a tutorial at the 23rd International European Conference on Parallel and Distributed Computing (Euro-Par) 2017, August 2017, Santiago de Compostela, Galicia, Spain.

"CPUs, GPUs, FPGAs: Managing the alphabet soup with Intel Threading Building Blocks," Michael Voss, Pablo Reble and Jackson Marusarz, a tutorial at the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP) 2017, February 4, 2017, Austin, TX, USA. http://ppopp17.sigplan.org/event/ppopp-2017-tutorials-cpus-gpus-fpgas-managing-the-alphabet-soup-with-intel-threading-building-blocks with code samples available at https://github.com/01org/tbb/tree/tbb_2017_PPoPP17

There is improved support for heterogeneous programming planned for the Intel® Threading Buildings 2018. Any new features and improved support will be reflected in the tutorial content and lab exercises. In addition, James Reinders will join as a presenter to provide additional content on Intel TBB and its relevance for HPC and heterogeneous computing.

3. Tutorial Outline (half-day, with full-day option hands-on)

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
- Relevance of TBB for HPC and heterogeneous computing
- 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 hybrid pipeline and parallel_for implementations
- An overview of experimental evaluations

4. Hands-on Part

A similar Tutorial has been previously presented at 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming and will be presented at 23RD International European Conference on Parallel and Distributed Computing.

Material is available on github (https://github.com/01org/tbb/tree/tbb_2017_PPoPP17) under the official TBB repository.

The general outline for the hands-on exercises at PPoPP 2017 was:

- "Hello TBB" verifying that the environment is set up correctly
- Implement small example as a shared-memory flow graph
- Adding async_node to the example to overlap an asynchronous computation
- Adding streaming_node to use an OpenCL-compatible device (GPU)
- Targeting an FPGA

Step-by-step instructions were provided to attendees and they were lead through these steps by the instructors. Feedback from PPoPP 2017 was very positive about the hands-on part of the tutorial.