Task Sharing: Collaborative CPU-GPU Execution

Michael Boyer, Kevin Skadron
University of Virginia Department of Computer Science
boyer@cs.virginia.edu
SRC Tasks 1607.001 and 1972.001

Abstract
In recent years, there has been growing interest in leveraging the enormous computational power of graphics processors for accelerating general-purpose applications. This generally requires decomposing an application into serial and data-parallel phases, with the parallel phases expressed as kernels. Typically the GPU is idle during the serial phases of the program while the CPU is idle during the data-parallel kernels. For kernels in which the two devices provide comparable performance, performance can be improved by allowing the two devices to work collaboratively. We are currently developing a software framework, based on OpenCL, to automatically divide a kernel’s execution across the CPU and GPU with minimal programmer effort.

OpenCL
OpenCL is a programming model for heterogeneous systems that is supported by more than 30 companies, including SRC member companies AMD, Freescale, IBM, Intel, and TI.

Serial code executes on the host device (CPU) while data-parallel kernels can be executed on other devices in the system (GPU, DSP, or even on the CPU) The programmer only needs to write one version of the kernel, rather than a different version for each device.

Executing an OpenCL kernel involves the following steps:
1) Choosing a device.
2) Compiling the kernel for that device.
3) Transferring input data from the host to the device.
4) Executing the compiled kernel on the device.
5) Transferring output data from the device to the host.

Preliminary Results
Application: HotSpot

Future Work
- Finish implementing task sharing framework.
- Measure performance impact on a diverse set of applications.
- Compare different algorithms for kernel division.
- Extend framework to support more than two devices.

Motivation
Theoretical performance:

Data Access Patterns
For iterative applications, if we cannot determine data access patterns statically, then the CPU and the GPU must have a consistent view of memory before each kernel launch.

Preliminary Results
Application: HotSpot

Future Work
- Finish implementing task sharing framework.
- Measure performance impact on a diverse set of applications.
- Compare different algorithms for kernel division.
- Extend framework to support more than two devices.

Related Work
Qinn [1] and Merge [2] allow a programmer to distribute computations across multiple heterogeneous devices. Both systems require the programmer to write different code for each device. Merge is restricted to map-reduce style applications.

Same Program for All Processors (SPAP) [3] is a programming language that can be mapped to multiple devices in a heterogeneous system. SPAP requires the programmer to either use a small set of language constructs that have already been implemented on the supported devices, or implement a new operation separately for each device.

CUDAAS [4] extends NVIDIA’s CUDA programming language to allow a kernel to be executed across multiple GPUs. CUDAAS requires the programmer to explicitly divide the work among the multiple devices.


Acknowledgments
This work is supported by an SRC GRC Fellowship. SRC grants 2007-HJ-1607 and 2008-HJ-1172, and equipment donated by AMD and NVIDIA. We would also like to thank Chris Gregg and Mariabel Guevara for their helpful feedback.

Technical Transfer
Liaison: Srilatha Manni (AMD)

Biography
Michael Boyer was awarded the Bachelor of Science Degree in Computer Engineering from Union College in 2006. As an undergraduate he received several awards, including the Loughry Prize for outstanding junior project in Computer Engineering. He participated in the pilot for the SRC Undergraduate Research Assistants Program, presenting his research at the Graduate Fellows Conference in 2004. He has also completed three internships with Intel. He is currently a Ph.D. student at the University of Virginia, where he is researching software support for heterogeneous architectures under the guidance of Kevin Skadron. He is supported by an AMD/Maruboo Khan Fellowship from SRC.