Short Modules for Introducing Heterogeneous Computing*

Conference Tutorial

David P. Bunde[1], Apan Qasem[2], Philip Schielke[3]
[1]Knox College
Galesburg, IL 61401
dbunde@knox.edu
[2]Department of Computer Science
Texas State University
San Marcos, TX 78666
apan@txstate.edu
[3]Concordia University Texas
Austin, TX 78726
Philip.Schielke@concordia.edu

Abstract

CS faculty have spent the last several years adding parallel computing to their curricula since essentially all processors sold today have multiple cores. A typical target system is a multicore processor with identical cores. This is the configuration for most current desktop and laptop systems, but the technology continues to evolve and systems are incorporating heterogeneity. Many phone processors include cores of different sizes so the phone can vary its power and performance profile over time. Other processors incorporate low-power modes or instructions for specialized computations. Meanwhile, high-end systems make heavy use of accelerators such as graphics cards. We are at a stage where heterogeneous computing concepts should pervade the curriculum rather than being limited to upper-level courses.

This tutorial motivates heterogeneous parallel programming and then presents modules that introduce aspects of it such as ener-

gy/performance tradeoffs, SIMD programming, the benefit of memory locality, processor instruction set design tradeoffs, and CPU task mapping. Each module uses only a few days of class time and includes assignments and/or lab exercises which are available online (https://github.com/TeachingUndergradsCHC/modules/). Here are the modules:

1. The first module shows the challenges and benefits of task mapping on a heterogeneous system. The module includes a lab to provide students with hands-on experience running parallel workloads in heterogeneous environments. It is aimed at CS 2, but also fits in Systems and Parallel Programming courses.

2. The second module looks at heterogeneity on ARM processors, particularly Thumb mode, a low-power mode with restricted instructions. The module is based on the Raspberry Pi, a low-cost system aimed at hobbyists. It highlights performance/power tradeoffs and is aimed at Computer Organization.

3. The third module shows how memory locality can improve performance on a program that uses CUDA to run on a graphics processing unit (GPU). This module demonstrates heterogeneity resulting from both CUDA's SIMD model of computing and the different memory types on a GPU. It highlights memory locality and is aimed at systems-oriented courses.

Acknowledgements

This tutorial presents work supported by NSF grants OAC-1829644 & OAC-1829554.