Heterogeneous Computing for Undergraduates: A Module-Driven Approach

Apan Qasem
Texas State University
San Marcos, TX
apan@txstate.edu

David P. Bunde
Knox College
Galesburg, IL
dbunde@knox.edu

Philip Schielke
Concordia University Texas
Austin, TX
Philip.Schielke@concordia.edu

ABSTRACT

To achieve aggressive performance and power goals, HPC systems are becoming increasingly heterogeneous. This represents an educational challenge since few current curricula include much about heterogeneous computing except possibly in upper-division electives. This poster presents a set of modules to integrate heterogeneous computing concepts in an undergraduate computer science curriculum. We discuss coverage of topics, the design philosophy and our experiences implementing the modules at different institutions. The modules are being developed as part of an NSF-funded initiative to enhance the Advanced Cyberinfrastructure workforce. The poster also discusses the synergistic activities undertaken in the broader project.

1 INTRODUCTION

The need to increase performance per watt, coupled with the demands of processing diverse workloads, has triggered an industry shift towards heterogeneous computing systems. Integration of high-performance CPUs with energy-efficient GPUs is now common in HPC. Architectural heterogeneity has also permeated mobile processing, cloud computing, and the Internet of Things. Machine learning practitioners routinely use accelerators in both training and inference. Given this pervasive heterogeneity, it is imperative that we teach CS graduates about these complex systems.

Current undergraduate computer science (CS) curricula have yet to catch up with this phenomenon and lacks sufficient coverage of heterogeneous computing (HC). Heterogeneity is covered only as an upper-level elective and that too primarily at R1 institutions. Because the CS/CE curriculum is already over-crowded, adding a new required course in HC is not feasible without increasing the time to graduation. Furthermore, HC spans different areas such as programming, algorithms, and architecture so it is better covered in multiple courses rather than a single course.

This poster presents a collection of teaching modules to introduce heterogeneous computing to computer science undergraduates. The modules and associated materials are freely available to educators. The poster provides a high-level overview of the modules and then briefly discusses the design philosophy, our experience implementing them in the classroom, and out-of-class synergistic activities.

2 MODULE-BASED INTEGRATION

We have identified a set of modules that collectively provide broad coverage of HC. Table **??** lists the modules along with the courses in which we recommend they be introduced. The modules cover both hardware and software, spanning computer architecture, programming models, and algorithms. Each module is designed with a set of learning outcomes. The strategy is to develop modules that provide broad coverage of HC topics and inject them into several courses across the curriculum in a way that creates significant coverage of HC concepts along every path through the major.

The modules are not focused on teaching a specific language for programming heterogeneous systems but rather cover a broad spectrum of HC topics. The goal is to expose *all* CS undergraduates to HC and not just to graduate a few experts with specialized knowledge of a single platform.

3 DESIGN PRINCIPLES

Our goal is to integrate HC content into the curriculum using the early-and-often approach. To facilitate integration and adoption, each module's design follows four basic principles.

(i) Abstraction: Module topics are selected based on the level of abstraction at which they can be introduced. Topics that can be explained at a higher level of abstraction are introduced early in the curriculum. For example, Amdahl's Law and its performance impact on heterogeneous systems is introduced in a course module directed at CS 1 students.

SC '21, Nov 14–19, 2021,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Partial list of available HC modules across six categories.

Category	Name	Sample Topics	Course
Fundamentals	Elementary Notions	concurrency, parallelism, and energy efficiency	CS1, CS2
	Task Mapping	task mapping, workload characterization	CS2, OS
	Pollack's Rule	performance modeling, Amdahl's Law	Org, Systems
Algorithms	Hybrid Algorithms	accelerator offloading; complexity analysis	Algorithms
	Algorithmic Design Patterns	reduction, scan, stencil	Parallel Prog
Architecture	Introduction to ARM	performance/power tradeoff, vector instructions	Org, Arch
	CUDA Memory Hierarchy	blocking/tiling optimizations	Arch, Org
Programming Models	CUDA	accelerator programming, SIMD	Systems, Org.
	OpenMP	SIMT, target offloading in OpenMP	Systems, Org
Performance	Amdahl's Law	impact on sequential, parallel and heterogeneous programs	Arch.
	Energy Efficiency	power vs. performance trade-off; use of DVFS	Arch

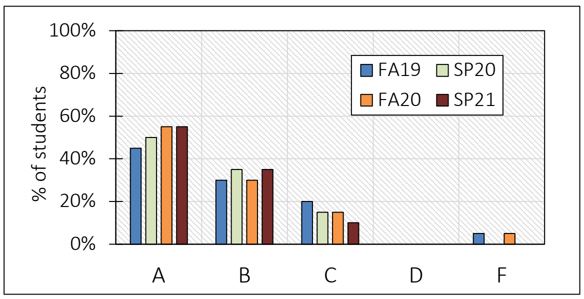

Figure 1: Student grades for learning outcomes assessment. Modules A1 and A2, Fall 2019 - Spring 2021

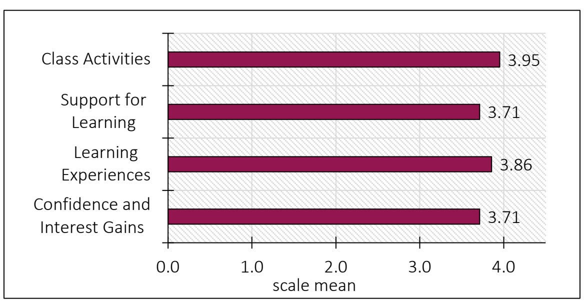

Figure 2: Student perception. Modules A1 and A2, Fall 2019 - Spring 2021

(ii) Context: Modules take advantage of heterogeneous context. For instance, when covering process scheduling in an OS course, issues related to *task offloading* can be discussed. When context is exploited, modules can simply extend topics already being covered, minimizing disruption.

(iii) Active Learning: Modules emphasize active learning. They include hands-on activities, in-class and self-paced, and make heavy use of Jupyter Notebooks and Google Colab.

(iii) Adoption: Modules are short (1-3 contact hours) and self-contained. They include lecture notes, assignments, exam questions, homework problems and solutions, and pedagogical notes to facilitate adoption.

4 EXPERIENCE

The listed modules have been implemented in multiple institutions and several more are currently being implemented. Fig. **??** shows assessment results from two of the Fundamentals modules introduced in CS1 and CS2 classes. In each semester, at least 85% of the students received a passing grade. Student performance improved over time, primarily as a result of the faculty having more experience.

In each instance when a module is implemented, we conducted an end-of-the semester survey to gauge student interest in the topic and assess student perception of the learning experience. Fig. **??** shows the average survey results for the same two modules. Students answered each question on a scale of 0-4 (e.g., strongly disagree, disagree, neutral, agree and strongly agree). Overall, the students rated the learning experience and instructional environment very positively.

5 BEYOND MODULES

For successful implementation of the modules, we have initiated several out-of-class activities. We discuss two here.

Industry partnerships. To ensure that the modules are relevant for future professionals, we have partnered with experts from Intel, AMD, NVIDIA, and Qualcomm. They provide suggestions on the design of modules and review the created modules.

Faculty Training. We identified faculty expertise as a primary obstacle in integrating HC content. To address this issue, we have undertaken a faculty training program. Faculty from different types of institution take part in a week-long workshop in which they receive training on teaching the modules. There is further follow up session for the participants to discuss their experiences in teaching the modules.

6 CONCLUSION

This poster is being submitted under the HPC Education category. Our hope is that during the poster session, we can engage with the HPC community to get feedback to improve and extend the module collection.