Early Adoption - Parallel Computing: Keys to a Future in Computing

Stephen Providence
School of Science
Hampton University
Hampton, Virginia 23668–0001
Email: stephen.proviencehamptonou.edu

Abstract—The triuvirate: parallel programming, parallel architecture and parallel algorithms is the mantra for education and research in parallel computing [2]. These three are necessary subareas that will prepare an undergraduate srudent to pursue studies in computational science or to develop software systems that exploit parallelism to solve NP-hard problems. We have been developing a curriculum that will take UG students in their freshmen year up through the masters level in parallel computing. The goal is to prepare students for the future of computing and to eventually institute a doctoral program in computational science or computer science.

I. Introduction

There is strong interest in restructuring the standard serial computing curriculum of computer programming, computer architecture and computer algorithms into a hybrid of both sequential and parallel versions of the three subareas of PDC. There is a strong belief that we can accomplish such a task at Hampton University [4], a university that is not well known for computational science and computing science.

II. Courses Impacted

The following are course descriptions as per syllabi and the University catalog. These are the courses that are under my instruction and are offered for Spring 2011, where one prerequisite is impacted:

1) CSC 395-02 (ParProg) - Special Topics: Supercomputing - Parallel processing models and architectures. Concurrent processes and controls. Parallel programming, algorithms and architectures. Problem solving on clusters. Prerequisite: one semester computer programming in a high-level language and permission of the instructor.
2) CSC 205-01 (Systems2)- Computer Architecture, Organization and Systems II - Intermediate logic design including truth tables, logic diagrams, Boolean functions and Karnaugh Maps. Computer architecture including CPU design, memory organization, I/O processing including programmed I/O, interrupt I/O and direct memory access. Coding. Prerequisite: CSC 204.
   a) CSC 204 (Systems1) - Computer Architecture, Organization and Systems I - Binary number representation of arithmetic. Computer structure. Addressing techniques. Storage allocation. Subrouting linkage. Relocatabiility and program segmentation, bit manipulation. Operating system supplied I/O routines and interface using a systems programming language and assembly language. Macros. Prerequisite: CSC 152 - Computer Programming II, MAT 117 - Precalculus Mathematics I.

   System1 and Systems2 are seen as two parts of a Systems core course.

III. Curriculum Topics

The Spring 2011 courses are the starting point focusing on parallel architecture. A broader view is in the first two years of undergraduate study, where we seamlessly will integrate parallel computing instruction into the following courses:

1) Parallel Programming in CSC 151, 152 (CS1, CS2) - Computer Programming I, II
   - Unix, RHEL and MacOSX usage and simple scripting and editing
   - CUDA [3] and OpenCL [1] programming
2) Parallel Architecture in CSC 204, 205 (Systems1, Systems2) Computer Architecture, Organization and Systems I, II
   - on-chip parallelism, shared-memory multiprocessors, Flynn's Taxonomy

- message-passing multicomputers, grid computing, multi-, many-core architectures, GPUs, FPGAs

3) Parallel Algorithms in CSC 251, 252 (DS/A1, DS/A2) - Data Structures and Algorithm Analysis I, II
   - Ian Foster's Design Methodology - partitioning, communication, agglomeration and mapping
   - case studies: Boundary Value Problem, Finding the Maximum, n-Body Problem, benchmarking, performance

DS/A1 and DS/A2 are seen as two parts of a DS/A core course.

IV. Evaluation Plan and Topic Integration

The following outlne will assist in course evaluations:

1) Curriculum Committee translates proposed Learning Outcomes → Course Objectives and Expected Outcomes
   - examine metrics and qualify them according to University, School and Department policies and mission statements
2) instructors measure Expected Outcomes vs. Student Performance via assessments of:
   - group/team projects and examinations
3) take student surveys at mid-semester and the end of each semester
4) find evidence of successful integration of sequential and parallel computing
5) instructors measure survey results against Expected Outcomes
6) Curriculum Committee makes changes to course, instruction methods and course materials
7) repeat step 2

V. Conclusion

Partnering with industry is paramount. Intel^{tm} and NVIDIA^{tm} have strong interest in furthering the success of their multi- and many-core designs, respectively. Ultimately, these and other industry leaders will judge the efficacy of this effort by hiring our graduates. Additionally, these vendors are ready to supply hardware and software resources in support of this effort.

Parallel computing is here to stay since multiple processors are widely available in relatively inexpensive. Both the public and private sectors require expertise to exploit this reenergence of parallelism. This plan will prepare our students to take advantage of multi- and many-core microarchitectures. They will view parallel from three sides and will be capable of conducting research in related areas. We expect that our graduates will manage and direct the future computing landscape.

References

[1] D.B. Kirk, W.W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach, Morgan-Kaufmann, 2010

[2] S. Prasad, NSF/TCPP curriculum on Parallel and Distributed Computing - Core Topics for Undergraduates, NSF White Paper, December, 2010

[3] J. Sanders, E. Kandrot, CUDA by Example: An Introduction to General Purpose GPU Programming, Addison-Wesley, 2011

[4] A.B. Shiflet, Computational Science in a Liberal Arts College, JCSC, Consortium for Computing Sciences in Colleges, 18, 2, 165-168, December, 2002