Concurrent CS: Preparing Students for a Multicore World

Daniel Ernst
Assistant Professor
Department of Computer Science
Eau Claire, WI, USA 54702-4004
dernst@uwec.edu

Daniel Stevenson
Associate Professor
Department of Computer Science
Eau Claire, WI, USA 54702-4004
stevende@uwec.edu

ABSTRACT

Current trends in microprocessor design are fundamentally changing the way that performance is extracted from complex systems. The promise programming models of sequential uniprocessor execution is being replaced quickly with a need to exploit parallelism. This shift has generated a need for new approaches to teaching computer science. Although many universities have developed a set of methodologies to support their students in effectively managing multicore workloads as part of a freshman design experience [1], the majority of students are not learning the concurrency concepts that are becoming increasingly important in ensuring the short-term and long-term success of our computer science (CS) graduates. This shift in the current CS workforce is not yet equipped to handle this shift in hardware, and industry recognizes this. As part of their “Terascale Initiative”, Intel has put out a call to programmers and educators alike to expand their skill sets and curricula to include multiprocessor systems. Parallel programming practitioners in the past have been limited to a small, specialized community of computer scientists, as access to multiprocessor systems was only available to those working in the research departments of large corporations, universities, or national laboratories. Because of this, the curricula of most computer science departments have only taught parallel programming as an advanced elective course, if at all.

However, our current undergraduates students will likely spend their entire career working on multiprocessors. The current CS workforce is not yet equipped to handle this shift in skills, and industry recognizes this. As part of their “Terascale Initiative”, Intel has put out a call to programmers and educators alike to expand their skill sets and curricula to include multiprocessing. These concepts are becoming increasingly important in ensuring the short-term and long-term success of our students.

Our approach is to give students practice with the concepts behind parallel programming early and often by integrating them into our existing coursework. We have developed modules for CS1, CS2, and an algorithms course which introduce students to some of the basic concepts of parallelism in a way which is straightforward, interesting, and interwoven into other course topics. Instead of spending time on teaching students specialized parallel programming languages, we focus on having students learn and apply principles of concurrency within languages with which they are already comfortable, such as Java and its object-oriented use of threads. Our goal is to have students emerge from the early years in parallel programming tools, it is still necessary for students to be exposed to parallel programming early and often by integrating them into our curriculum design.

Concurrent CS: Preparing Students for a Multicore World

Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science, 2008

Modern computer processors have undergone a significant change in the last five years. All of the major vendors now sell “multi-core” processors almost exclusively. While these new machines are capable of tremendous performance, software developers need to write computer programs differently in order to gain any benefit. In this paper, Dr. Daniel Stevenson and I describe our efforts to bring parallel computing concepts into the mainstream of computer science education. We discuss basic guidelines for content and provide numerous example projects to enhance student learning of these topics.