AC 2009-1683: INCORPORATING PARALLEL COMPUTING IN THE UNDERGRADUATE COMPUTER SCIENCE CURRICULUM

Afsaneh Minaie, Utah Valley University

Reza Sanati-Mehrizy, Utah Valley State College

Incorporating Parallel Computing in the Undergraduate Computer Science Curriculum

Abstract

Parallel and distributed computing are subjects generally reserved for graduate programs. With the design of the multi-core architecture, it is essential that parallel design of software be integrated into the undergraduate computer science curriculum. Parallel programming represents the next turning point in how software developers write software. This paper will study different approaches that are used by different institutions of higher education around the world to integrate parallel computing into their curriculum.

Teaching parallel computing concepts to undergraduate students is not an easy task. Educators need to prepare their students for the parallel era.

Introduction

A fundamental technique by which computations can be accelerated is parallel computation. The main reason for executing program instructions in parallel is to complete a computation faster. However, majority of programs today are incapable of much improvement through parallelism, since they have written assuming that instructions would be executed sequentially[1]. Since sequential computer performance has increased for decades; parallelism has not been a significant part of programming. Improvements in the sequential performance have been due to the combination of technological improvements and the incorporation of parallelism into sequential processors. Hidden parallelism with increasing clock speeds has allowed each succeeding generation of processor chip to execute programs faster, while maintaining the delusion of sequential execution[1]. Given current technologies, sequential program execution is approaching its maximum speed. To be able to achieve performance improvements, we need programs that have multiple instruction streams that operate simultaneously. By the advent of the first multi-core chip in 2005, it was observed that the existing software cannot exploit multi-chip directly. Programs that cannot exploit multi-core chips do not realize any performance improvements. Another observation was that most programmers do not know how to write parallel programs. Programs need to change, and for that to happen, programmers need to learn how to write parallel programs. Parallel programs need to be scalable. A parallel program is scalable if it is capable of using progressively more processors. It is important to achieve scalable parallelism.

Multi-Core Processors

The computer industry is rapidly moving toward multi-core architectures also known as chip multiprocessors (CMP), where multiple cores can independently execute different threads. The computing platforms are being designed with microprocessors that have multiple execution cores on a single chip. It is predicted that the number of cores on a single chip is going to rapidly grow into tens, or even hundreds, of cores on a single chip. Moore's law [2] states that the number of transistors on a single chip should double every eighteen months. As the number of transistors doubled every eighteen months, so did the speed of the processors. However, after 2003 the speed of processors didn't keep up with the increase in number of transistors on the chip. Rather than faster CPUs, the computer industry now offers more CPUs per machine through the use of multi-core processors. As the speed of the processors continue to increase, so do their heat and power drain. By using multi-core architecture, the speed increases without the traditional drawbacks of faster processors, which include power consumption and heat dissipation. Multi-core architectures are used to enhance throughput and power efficiency of processors. Now the prediction is that the number of cores on a chip would double with each silicon generation[3.]

This change in computer architecture requires change in programming paradigm. The era of developers simply waiting for faster processors to save their slow performing applications is over. For developers to take advantage of this multi-core environment, they must learn to write software for tightly-coupled shared memory multiprocessor systems. Industry leaders are challenging the software developers to update their skills in order to effectively develop software for these new architectures. There is a school of thought that says that software design needs to keep up with the Moore's law; that is software should double the level of parallelism that it can support with every technology generation[4].

Parallel computing is not a new concept and has been extensively studied for many years. It has always offered the potential for scalable high performance computing but not achieved enough drive to get considerable exploitation in a mainstream application. Parallel computing is used in a wide range of fields, such as bioinformatics, economics, astrophysics, weather forecasting, and robotics. As parallel computers become larger and faster, it becomes possible to solve problems that previously took too long to run. In the past, parallel programmers have been limited to a small community of computer scientists working in research departments of large companies, universities, or national labs. Therefore, the undergraduate curricula of the majority of computer science departments did not include any parallel computing courses. Some programs offered parallel computing as an advanced elective course. With the design of the multi-core architecture, undergraduate computer science students will likely spend their entire career working on multiprocessor machines. Teaching parallel computing concepts to undergraduate students is not an easy task. To prepare our students for parallel programming, it is essential that parallel design of software be integrated into the undergraduate Computer Science curriculum. Parallel programming represents the next turning point in how software developers write software[9]. In the Computer Science Curriculum 2008 (An interim revision of CS 2001), within Recent Trends section, there is a section on the growing relevance of concurrency which says that

"The development of multi-core processors has been a significant recent architectural development. To exploit this fully, software needs to exhibit concurrent behavior; this places greater emphasis on the principles, techniques and technologies of concurrency.

Some have expressed the view that all major future processor developments will include concurrent features, and with even greater emphasis on the concurrency elements. Such a view implies the increased emphasis on currency will not be a passing fashion but rather it represents a fundamental shift towards greater attention to concurrency matters.

The increased ubiquitous nature of computing and computers represents a further factor in heightening the relevance of this topic and again there is every indication that this will only gain further momentum in the future. It is expected that these observations will have implications for many knowledge areas in the future curriculum guidelines." It is vital that parallel computing concepts be integrated into undergraduate computer science curriculum. Educators need to prepare their students for the parallel era.

Parallel Computing Courses in China

A majority of universities in China have integrated parallel computing concepts into their curriculum[7]. These universities have taken two different approaches for integrating parallel concepts in to their programs. One approach has been to offer a new course on parallel processing. The other approach has been to alter the content of an existing course. Following is a list of traditional courses that have been altered to introduce mutli-core processing concepts:

- Computer Organization
- Computer Architecture
- Operating System
- Pervasive Computing
- Embedded Systems
- Real Time Systems
- Undergraduate Research Courses

Table 1 lists universities in China that have altered their traditional courses to include the multi-core experience for their students. From this data, it can be seen that Computer Architecture, Computer Organization, Operating Systems, and Embedded Systems are courses that have been modified.

Universities	**Computer Organization**	**Computer Architecture**	**Operating System**	**Embedded Systems**	**Programming**	**Real Time**	**Research Course**
Dalian University	X		X	X			
East China National University	X	X					
Shanghai Jiao Tong University	X		X				
Zhejiang University	X	X	X				
Xi'an University of Technology	X	X		X		X	
Northeastern University	X						
Tsinghua University		X					X
Beihang University		X		X			
Beijing University		X	X				
Lanzhou University		X					
Fudan university		X	X		X		
Southeastern University		X					
University of							

Science and Technology of China		X	X				
Hauzhong University			X				
Beijing Institute of Technology				X			
Tongji University							X
Nankai University				X			

Table 1: Universities in China and the Traditional Courses that were altered[7]

Parallel Computing Courses in United States

The University of Wisconsin – Eau Claire

The approach which is taken at the University of Wisconsin – Eau Claire in order to prepare their students for the parallel Computing experience is to give them practice with the concepts behind parallel processing early and integrating them into their existing courses[5]. They have integrated parallel computing concepts into three of their courses: CS1, CS2, and Algorithm. In CS1, they focus on the decomposition step, and in CS2, they begin to introduce a bit more orchestration of concurrent processes. Their Algorithm class is taken in the sophomore year. They present more complex issues in parallelism in this course. The approach that they have taken is to introduce the parallel programming concepts slowly through three of their courses.

Longwood University

In CS1 and CS2, parallel paradigms are mentioned and demonstrated with use of the threads library; however, no actual assignments are given. In their CS3 (Data Structures) class they look at parallel versions of some algorithms. They offer a separate course dedicated to parallel programming using PVM and MPI[6].

Table 2 provides a study of parallel computing course offerings of selected universities in the world. In the United States, course offerings of seventeen universities were studied. From this study it can be seen that universities are using four models to integrate parallel computing into their computer science curriculum:

1. Offering an undergraduate course on Parallel Computing
2. Offering a graduate course on Parallel Computing
3. Integrating parallel Computing concepts into their traditional courses
4. Combining model 1, 2, and 3

From the table it can be seen that Massachusetts Institute of Technology is using model 4 and they are offering an undergraduate course, a graduate course, and have integrated parallel concepts in their traditional courses. On the other hand the majority of liberal art schools do not use any of these models and parallel computing concepts are not being taught in their undergraduate computer science programs.

Universities	Undergraduate Parallel Computing Course	Graduate parallel Computing Course	Parallel Computing Concepts as a Module in other Classes
Massachusetts Institute of Technology	Applied parallel Computing	Theory of parallel Systems Theory of Parallel Hardware	6-827 Multithreaded Parallelism: Languages and Compilers 6-884 Complex Digital Systems
Berkeley	No	Parallel Processors CS267 Application of Parallel Computers	252 – Graduate Computer Architecture 266 – Introduction to System Performance
Harvard	No	262 – Introduction to Distributed Computing CS 355, 356 – Computational Complexity	CS 246 – Advanced Computer Architecture CS260r – Topics in Computer Systems
Stanford	No		CS 212- Computer Architecture and Organization CS312A: Advanced Processor Architecture CS240D: Distributed Storage Systems CS321- Information Processing for Sensor Networks
University of Utah	No	CS 6230 High Performance Parallel Computing	
Carnegie Melon University	85-419 Introduction to Parallel Distributed Processing		15-740 Computer Architecture 18-741 Advanced Computer Architecture 15-845- Mobile and pervasive Computing 15-745 – Optimizing Compilers for Modern Architecture
United States Military Academy	No	No	
India Institute of Technology	No		Yes
University of California Davis	CSE160-Introduction to Parallel Computing	CSE225- High Performance Distributed Computing CSE240B – parallel Computer Architecture CSE 260 – Parallel Computation CSE 261 – Parallel and Distributed Computation	
University of Pennsylvania	CIS 434 – Into. To Parallel and Distributed Programming		CIS 371 – Computer Organization and Design
Purdue University	No	CS 52500 – Parallel Computing	
University of North Carolina at Chapel Hill	No	COMP 633 – Parallel and Distributed Computing	
John Hopkins University	600.320 – Parallel Programming	600.420- Parallel Programming	
Polytechnic University	CS 3254 – Introduction to Parallel and Distributed Systems CS 342 – Algorithms for Parallel and Distributed Systems	CS 6273 – Performance Evaluation of Computer Systems	
University of Illinois at Urbana-Champaign	420 – Parallel Programming: Science and Engineering		
Sharif University of Technology, Iran	40-647 – Parallel Processing		
Tsinghua University, China	Introduction to parallel Programming		Advanced Computer Architecture
Peking University, China	Parallel Programming		
University of Science and Technology of China	Parallel Computing		Computer Architecture Operating System
Northwestern Polytechical University, China	Parallel Computing		Distributed Computing System
Harbin Institute of Technology,			Distribution System

China			
National University of Deference, China	Multicore and Multicore Programming		
Zhejiang University, China	Parallel Computing and Multicore Programming	Multicore Computing	Computer organization Computer Architecture Operating System
East China Normal University	Intel Multicore Technology		
Shanghai Jiao Tong University, China	Multicore Systems and Programming		Computer Organization Operating Systems
Southeastern University, China	Multicore Technology and Multicore Programming		Computer Architecture
South China University of Technology	Multicore Software Design		
Wuhan University, China	Multicore Architecture and Programming		
Sun Yat-sen University, China	Multicore Technology and Optimize Programming		
Northeastern University, China	Multicore Programming		Computer Organization
University of Karlsruhe, Germany		Multicore Software Engineering	
Georgia Tech University		Distributed &Parallel Systems	
Frederick University, Cyprus	ACOE401 – parallel Processing		
University of Wisconsin – Eau Claire			CS1 CS2 Algorithm
Longwood University	parallel programming using PVM and MPI		CS1 CS2 CS3
Utah Valley University	No	No	No
IONA School of Arts and Science	No	No	
Grove City College	No		
Quinnipiac University	No		
Shenkar College[10], Israel	Introduction to Parallel Computing		

Table 2: A Survey of Universities with Regard to Parallel Computing Offerings

Summary and Conclusion

Integration of parallel programming concepts in the undergraduate computer science curriculum has started in many universities worldwide. For example, at Shenkar College of Engineering and Design in Israel, a new course called Introduction to Parallel Computing has been developed. The university is developing an advanced course on the topic[10]. A majority of universities in China have integrated parallel concepts into their curriculum[7]. In the United Sates, there are some universities that are offering an undergraduate course on parallel computing; however, the majority of liberal arts universities have not integrated the parallel processing concepts in their curriculums yet.

When should the parallel computing concepts be introduced into the computer science curriculum? Some computer scientists believe that because of the complexity of parallel computing, it should be introduced as early as CS1. They believe that the concepts should be introduced slowly as modules in different traditional courses. By introducing the parallel computing concepts early in the program, students always have a parallel option as a solution to problems that they want to solve. Some believe that it should be offered as a senior level required course. The second option might be harder to implement since adding a new course to the curriculum is not an easy task, as often times eliminating another course would be necessary. Offering it as an elective course is not a good option either since every student is not going to get this experience. It seems that adding the concepts slowly as modules to existing courses is a good solution for integrating the parallel computing concepts into the computer science curriculum.

As the computing industry rapidly moves toward multi-core and parallel processing architectures, tomorrow's computer scientists must be educated on the tools and methodologies for parallel computing. As educators, teaching parallel hardware and software today is vital to giving our students the tools they need to build tomorrow's hardware and software. It is crucial that parallel and distributed computing topics be integrated into computer science curricula.

References:

[1] Lin, Calvin and Lawrence Snyder, "Principles of Parallel Programming". Pearson Publishing Company, 2008.

[2] Moore, G.," Cramming more Components onto Integrated Circuits". Electronics 38, 8, 1995.

[3] Brin, S., and L. Page, "The Anaotomy of a Large Scale Hypertexual Web Search Engine". Technical Report, Stanford University, 1997.

[4] Fried, I. Intel: Software Needs to Heed Moor's Law". CNET News, May 2006.

[5] Ernst, Daniel and Daniel E. Stevenson, "Concurrent CS: Preparing Students for a Multicore World". ITiCSE'08, June 30 – July 2, 2008, Madrid, Spain.

[6] Graham, Integrating Parallel Programming Techniques into Traditional Computer Science Curricula, SIGSCE Bulletin, Volume 39, Number 4, December 2007.

[7] Chen, Tianzhou, and Qingsong Shi, "Multicore Challenge in Pervasive Computing Education, The 3rd International Conference on Grid and Pervasive Computing, IEEE, 2008.

[8] http://www.acm.org/education/curricula-recommendations , CS2008 Curriculum Update (draft), Accessed on January 30, 2009.

[9] Marowka, A. "Parallel Computing on Any Desktop", Communication of ACM, Vol. 50, no. 9, pp. 74-78.

[10] Marowka, Ami, "Think parallel: Teaching parallel Programming Today, IEEE distributed Systems Online, accessed on January 12, 2009.