Extended Abstract: Experiences of Teaching An Undergraduate Parallel Computing Course

Bo Hong
School of Electrical and Computer Engineering
Georgia Institute of Technology
bohong@gatech.edu

This article presents experiences in teaching an undergraduate parallel computing course. This is a new parallel computing course (Introduction to Parallel Computing) for senior ECE students at Georgia Tech. This is for the first time that parallel computing is systematically exposed to undergraduate ECE students at Georgia Tech. Students coming to the course have been prepared with knowledge of programming and computer architectures, but such preparation is almost exclusively limited to uni-processors and sequential programming. On the other hand, to provide a comprehensive overview of the field of parallel computing, the course expands over a rather broad range of topics including parallel computer architectures, parallel programming models, and parallel algorithms. The gap between the students' background and the course requirement makes the course challenging for both the students and the instructor. This article presents my teaching experience of the course, and my thoughts on how to make parallel computing friendly and accessible for undergraduate students.

For an undergraduate student with background in sequential computation only, parallel computing is difficult to understand because of two questions: (1) where does the previous knowledge of sequential computation apply? (2) how is a parallel program executed? Students will accept parallel computing as a *norm* when the two questions are answered, and can subsequently proceed with more abstract studies such as parallel algorithms.

1) *Making connections to the sequential experiences*. Students would be amazed when they find out that parallel programs consist of multiple processes/threads, and each one of the processes/threads is executed in exactly the same way as a regular sequential program. The coverage of my course starts with simple examples showing the decomposition of a parallel program, and the execution of the individual threads. The discussion then proceeds to the interesting part of data sharing, especially how is it implemented under various programming models. And the discussion then leads to the more subtle 'side-effect' problems of memory consistency and data races. With such a gradual shift of topics, the students can make connections to their sequential computation experiences, and realize that parallel computing has its root in sequential computation. Such an understanding will makes it much easier for the students to pick up more advanced concepts in parallel computing.
2) *Close the gaps between concepts and implementations*. For undergraduate students, parallel computing is challenging because the coding of a parallel programs does not easily translate to how it is executed. This will cause difficulties in the comprehension of parallel computing - the students (equipped with sequential computing concepts) are often perplexed by simple questions such as "how exactly will a multicore computer execute this parallel program?". Such confusion will make it more challenging for the students to accept the relatively abstract topics such as parallel algorithms. Multiple conceptual gaps are explained in detail in my course, such as how multiple threads share an address space, how does the operating system schedule the threads, how does the parallel architectures execute the threads, and for message passing, how does the operating system support inter process communications, and how does a system launch a parallel program. Clarification of these questions greatly helps the students to understand the execution of parallel programs, and it served as the foundation for more advanced topics such as parallel algorithms and multi-threading in GPU computing.

With such gradual transition from sequential computing to parallel computing, the students are able to get used to the idea of parallel processing as they can make easy connections to what they already know. In my experience, understanding how parallel programs are executed is a very important step for the students to transit from sequential thinking to parallel thinking. Subsequent core topics (architectural, programming models, and algorithms) become natural once the students accept the concepts of parallel computing. And their learning experience is further enhanced through programming projects that covers various aspects of parallel computing.