Teaching Parallel Computing: Architectures, Algorithms, and Applications

Abstract—Parallel architectures are becoming an increasingly attractive option for obtaining high-performance from devices ranging from embedded systems to supercomputers. Therefore, it is critical that the next generation of computer engineers are well aware of the concepts, problems, and opportunities in parallel computing. This paper presents a Parallel Computing course that focuses on the "Three A's of Parallel Computing", namely the underlying architectures, algorithmic understanding, and the applications. The course focuses on a wide range of parallel computing paradigms and programming interfaces which is covered through specific projects. This comprehensive approach is successful based on student and industry feedback.

Index Terms—Teaching, Parallel Computing, Architectures, Algorithms, Applications.

1 Introduction

Parallel computing has historically been explored through multiple computers. However, parallel computing has become even more critical with the recent advancements in general-purpose multicore and manycore architectures and domain-specific manycore architectures. These new architectures have become an increasingly attractive option for obtaining high performance and low power consumption since getting higher but effective frequency out of single-processor designs has become more complex. As a result, new parallel computing architectures are widely available in the market and have been used as an attractive option for overcoming the barriers in processor design. As technology scales, the International Technology Roadmap for Semiconductors (ITRS) projects that the number of cores will drastically increase to satisfy the performance requirements of future applications [1].

Despite the many advantages of parallel computing architectures over uniprocessor architectures, one of the critical questions raised by many researchers is the effectiveness of these architectures [2], [3]. One aspect of this problem is the infancy of the software solutions targeting such architectures. In general, current programs, compilers, and software architecture techniques rely on the fact that only one core is running on the background [4]. Hence, it becomes tough to use the underlying processing power effectively. There are some initial attempts to target this problem; however, these techniques are still in their infancy [4].

Therefore, the next generation of computer engineers must know the concepts, problems, and opportunities in parallel computing. In general, computer science or computer engineering curricula aim to provide a balanced education in the design and analysis of computer software and computer hardware. According to IEEE Computer Society and ACM Joint Task Force on Computer Engineering Curricula [5]: "Computer engineers are solidly grounded in the theories and principles of computing, mathematics, and engineering, and apply these theoretical principles to design hardware, software, networks, and computerized equipment and instruments to solve technical problems in diverse application domains." However, this may not always be achieved due to continuously changing technology [6].

Furthermore, current language extensions, libraries, or tools may not always be sufficient to use in parallel platforms directly. This, in turn, enforces software designers to have a basic knowledge of the nature of parallel architectures, underlying network topologies, programming paradigms, and algorithms. Therefore, it is vital to include similar computing education in undergraduate and graduate computer engineering curricula.

This paper presents experiences in the parallel computing course with hands-on parallel programming tasks. The first part of the course focuses on the architectural features of parallel computing platforms, including processing units and communication networks. On the other hand, the second part of the course focuses on algorithmic details of parallel applications in different architectural settings. Specifically, algorithm design, optimization, and complexity analysis are covered. The last part of the course is about applications and parallel program development on different parallel programming paradigms.

Although the given assignments vary over the years, the focus is three folds. The first assignment focuses on the message-passing environment and its use, whereas the second focuses on collective communication operations. A shared memory parallel system implementation follows this. The last task in the course focuses on GPU programming to give a better understanding of accelerator use cases in heterogeneous environments. These tasks aim to turn the theoretical discussions on architecture and algorithms into real-life examples. This way, it is possible to enable hands-on experience in state-of-the-art parallel programming environments.

We believe this is a productive course setting as it provides students the theoretical knowledge and programming experience on interesting real-world problems. It is precious when learning different algorithms on different architectures with different properties in a limited time frame. It also enables tackling similar issues in other architectures, thereby providing "experiential learning".

This paper demonstrates how parallel computing topics

Table 1
Parallel Computing course syllabus.

Week	Topic
1	Introduction
2	Parallel Programming Platforms
3	Trends in Microprocessor Architectures
4	Communication Costs in Parallel Machines
5	Interconnection Networks
6	Basic Communication Operations
7	Collective Communication Operations
8	Principles of Parallel Algorithm Design
9	Parallel Algorithm Models
10	Analytical Modeling of Parallel Programs
11	Parallel Programming Concepts
12	Programming Using the Message Passing Paradigm
13	Programming Shared Address Space Platforms
14	GPU programming
15	Performance Optimizations for Parallel Platforms

can be introduced along with different parallel computing concepts. Specifically, this course has been offered almost every year since 2012 as an elective course named "Parallel Computing" at Bilkent University, Turkey [7]. This course helps junior, and senior undergraduate students understand the hardware/software issues related to parallel computing and enable student participation through multiple projects and continuous discussions.

The remainder of this paper is structured as follows. The following section describes the course and its use in teaching parallel computing concepts. Section 3 gives the details about the parallel computing assignments, while conclusions are presented in Section 4.

2 Parallel Computing Course Overview

The Parallel Computing course at Bilkent University was first offered in the Spring of 2012, with an initial enrollment of 17 undergraduate students, who were all Computer Engineering majors. Over the years, the number of registered students reached 45, making the course among the most popular elective courses in the department. This course covers the "Three A's of Parallel Computing," underlying architectures, algorithmic understanding, and applications. The course is designed to give these general concepts and hands-on experience on related platforms.

Experiential learning describes the learning undertaken by students who can acquire and apply knowledge, skills, and feelings in an immediate and relevant setting [8]. This course aimed to provide such an environment. Instead of a passive environment, the Parallel Computing course seeks to achieve continuous in-class discussions on projects, providing students with an experiential learning environment. Students can react to lecture material from their personal experiences and apply the course material to real-life situations and new problems.

The overall organization of the course for a 15-week semester is shown in Table 1. There are three weekly lecture sessions where technical discussions and project meetings are held. Project discussions start at the beginning of the semester to set the stage correctly.

3 Parallel Computing Assignments

This section presents the four projects given as part of the course to cover the topics of Architecture, Algorithms, and Applications. The first two projects are implemented using MPI, whereas the third and the fourth projects are implemented on OpenMP and CUDA, respectively.

The first project aims to teach the basic MPI commands and distributed parallel computing through a task named *Compute Sum*. The objective of the problem is to write a serial, then a parallel program that takes an array of integers as input and prints out the sum of the elements in the file.

On the other hand, the second project focuses more on data decomposition and algorithms along with architectural aspects. More specifically, parallel decomposition based on partitioning the input and output data has been considered for *Quicksort*. In terms of architectural aspects, students were asked to implement the algorithms on a Hypercube.

The last two projects target the parallelization of convolution used in different applications. Students are asked to implement various versions of convolution using OpenMP in the third project. They extended this implementation in the fourth project and used CUDA and NVidia GPUs.

The projects asked students to analyze their implementations using state-of-the-art profiling tools, such as the NVProf and NSight. The students reported how their parallel implementation behaves regarding communication cost, expected speedup, etc. They provided reports with each project and reported results through plots for execution times/speedups with various inputs. Furthermore, they indicated their observations on the relationship between kernel size and constant memory implementations, memory behavior with changing block sizes, effects on the amount of data reuse, etc.

4 Conclusion

This paper presents a Parallel Computing course incorporating parallel architectures, algorithms, and applications. It gives details about how the course is organized and structured. Course assignments are given to explain the hands-on nature of the course. Since this course has been offered many times during the last decade with solid evaluations by both students and industry, we believe it is an effective way of teaching parallel computing.

References

[1] ITRS. International technology roadmap for semiconductors.

[2] Kumar, R., Tullsen, D.M., Jouppi, N.P., Ranganathan, P.: Heterogeneous chip multiprocessors. Computer **38**(11) (2005) 32–38

[3] R. Kumar et al. Single-isa heterogeneous multicore architectures for multithreaded workload performance. *ISCA '04: Proceedings of the 31st annual international symposium on Computer architecture*, (2004).

[4] Corezilla: build and tame the multicore beast. 2007.

[5] Joint Task Force on Computer Engineering Curricula, IEEE Computer Society and Association for Computing Machinery. Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering. *http://www.acm.org/education/curricula-recommendations,last visited on 12/09/2010*,2004.

[6] S. Ristov, N. Ackovska, V. Kirandziska and M. Gusev, Is the computer science curriculum ready to teach students towards hardwarizing?, 2016 IEEE Global Engineering Education Conference (EDUCON), Abu Dhabi, 2016, pp. 397-402.

[7] Cs 426 - parallel computing course, http://www.cs.bilkent.edu.tr/~ozturk/cs426/.

[8] Kolb. D. A. and Fry, R. (1975) 'Toward an applied theory of experiential learning;, in C. Cooper (ed.) Theories of Group Process, London: John Wiley.