

Eight Years Analysis of Adopting PDC in Data Structures at UCA

Bernard Chen

Department of Computer Science, University of Central Arkansas, Conway, AR, bchen@uca.edu

I. INTRODUCTION

University of Central Arkansas (UCA) is located at the center of Arkansas. The student population of UCA is between 10,000 and 12,000. The student body of UCA is around 20% minority and 60% female. UCA was established (in 1907) over a hundred years ago. It is the third largest public school as well as the third fastest enrollment growing public school in Arkansas.

Since 2011 fall semester, the Department of Computer Science at UCA has been selected as one of early adaptors for NSF/TCPP Curriculum Initiative on Parallel and Distributed Computing (PDC). Data Structures has adopted the parallel and distributed concepts for the past eight years. Our goal is to revise our adapted courses, share our successful experiences through this paper and attract more adopters.

II. INCORPORATING PDC TOPICS INTO DATA STRUCTURE

A. Introduction to adopted Data Structure class

Since 2011 Fall, Data Structures class has been regularly taught by Dr. Bernard Chen. According to Eileen Kraemer and our teaching experiences, “sorting” might be the best topic to introduce parallel concept in depth to CS major students in Data Structure class. As the result, during the sorting chapter, we (i) explain the parallelization of merge-sort wherein each level starting from bottom to top can be merged in parallel using $n/2$ processors thus requiring $O(2 + 4 + \dots + n/4 + n/2 + n) = O(n)$ time. Using $p \leq n/2$ processors will lead to $O(n/p \log(n/p) + n)$ time, hence $p = \log n$ is a cost-optimal choice. (ii) explain the parallelization of quick-sort using recursion. Compare the difference and the speedup potential of parallelization process between merge sort and quick-sort; (iii) discuss the difficulty of heap-sort parallelization. Summarize the $O(N \log N)$ sorting algorithms in both regular non-parallel and parallel versions; (iv) explain the parallelization of non-traditional sorting algorithms such as counting-sort and radix-sort.

B. Assessment

In the final project of the Data Structures class, students as a team (3 or 4 students) need to implement Insertion sort, Shell sort, Quick sort, Merge sort, Heap sort, Radix sort on a half million dataset to compare their performances. In addition, students are required to find an extra algorithm that has not been included in the list for implementation. Students are also required to explain how to parallelize the extra sorting algorithm with 2 and 4 CPUs in both oral and written presentation. A rubric has been used to evaluate student’s presentation in the concepts of the parallelization. Excellent means students are able to fully comprehend how to parallel

the extra sorting algorithm they found in both oral and written presentation. Good means students are able to present their ideas with minor flaws. Acceptable means students are able to use other sorting algorithm, such as merge sort, to implement the parallelization. Unacceptable means students are not able or forget to discuss the component of parallelization. Table 1 provides the rubric results of students’ performances on explaining how to parallelize the extra sorting algorithm for 11 semesters in 8 years. According to Table 1, more than half of the students (50.20%) received “Excellent” for their parallelization component. Almost 80% of the students received above 70% (Good + Excellent) of the evaluation. Only 2.85% (Unacceptable) of the students are failing in the process. Many students were willing to learn more about parallelization and have long discussion with the instructor even after classes. The final project gave students a great opportunity to digest what they learned about these new concepts. More importantly, most of the students expressed their willingness to take parallel computing related classes in the future so that they are able to enhance their ability to write parallel codes with parallel programming languages and libraries.

Table 1 The rubric results of students’ performances on explaining how to parallelize the extra sorting algorithm for 11 semesters in 8 years.

	#student	Excellent	Good	Accept	Unaccept
Fall 11	24	17	3	0	4
Spring 12	9	3	6	0	0
Fall 12	21	12	3	3	0
Spring 13	16	9	4	3	0
Fall 13	18	12	3	3	0
Spring 14	19	9	10	0	0
Fall 14	24	10	9	5	0
Spring 15	23	14	6	3	0
Spring 16	37	24	6	0	7
Spring 17	27	17	4	6	0
Spring 18	28	18	8	0	0
Total	246	145	62	23	11

III. CONCLUSION

Since multi-core computers are the major trend of current market, how to maximize those powers is an important issue to programmers. Computer Science students need to understand parallel computing, since it will become an integral part of their own careers. In this paper, we present our methods to adopt parallel and distributed computing into Data Structure curriculum. We also report our experience, methods and long-term evaluation results. We believe this article may provide useful insights to future NSF/IEEE TCPP early adopters and educators who also believe in parallel computing is a major component in current Computer Science curriculum.

Budget:

Items	Costs
Registration	\$525
Travel by driving Private owned vehicle (680miles * \$0.545/mile)	\$370.6
Lodging Per Diem (3 nights * \$135/night)	\$405
Meals & Incidentals Per Diem (3 days * \$64/day)	\$192
TOTAL	1492.6