Spring 12: IEEE-TCPP Curriculum Implementation Status at Ohio University

David W. Juedes
School of Electrical Engineering and Computer Science
Ohio University
Athens, Ohio, U.S.A.
Email: juedes@ohio.edu

Frank Drews
School of Electrical Engineering and Computer Science
Ohio University
Athens, Ohio, U.S.A.
Email: drews@ohio.edu

I. Introduction

This paper provides a status update on the implementation of the adoption of the NSF/IEEE-TCPP Curricular Standards for Parallel and Distributed Computing [1] at Ohio University. Our adoption of the NSF/IEEE-TCPP Curricular standards began in the Winter and Spring quarters of 2011-12, and has extended into the Fall and Spring of the 2012-2013 School year. This implementation is on-going and now involves a range of courses taught throughout the Computer Science and Computer Engineering curricula. In particular, our implementation involves four current required courses: CS 2401[1](CS/2) Introduction to Computer Science II , CS 3610[2](DS/A) Data Structures, CS 4000 Introduction to Distributed, Parallel, and Web-Centric Computing, and CS 4420[3](Systems) Operating Systems. These courses cover a broad range of material from the curricular standards on parallel and distributed computing.

This poster is organized as follows. Section 2 covers the evaluation strategy taken in this implementation. In particular, we discuss how we know whether each piece of the implementation is successful, and we discuss how we plan to determine whether we are placing the material in the correct location. In sections 3, 4, 5, and 6, we discuss the specific implementation status for each of the four courses mentioned above. Finally, we provide some insight into the placement of specific topics within the curricular recommendations with an example from our implementation experience.

II. Evaluation Plan

The implementation and evaluation strategy taken at Ohio University models our current outcomes based assessment strategy that we use for ABET accreditation purposes. We have chosen certain performance indicators for each module that can be repeated consistently across the curriculum to determine whether the approaches that we are taking are effective. In the case of the early adoption of the NSF/IEEE TCPP curricular standards, we seek to determine whether the placement of certain topics in certain courses is appropriate. As we will see in later sections, our performance indicators provide a clear indication that, sometimes, certain material may need to be repeated or assessed at multiple points in the curriculum to ensure that students achieve desired levels of competence.

As we discuss below, each topic is covered (roughly) in a "module" that may consist of one of more hours of coverage in lecture. The evaluation of each module involves four steps. The first step involves presenting the module and assigning a task for the students that matches the performance indicator. Such a task might be a quiz or a programming assignment, depending on the desired level of competence (K/C/A). The second step involves reviewing the student results, and based on a rubric, classifying student performance as either "excellent," "good," "developing," or "poor." We use a well-established goal that 70% of students achieve either "excellent" or "good" on the performance indicator. If this goal is not met, the performance indicator and assessment tool are re-evaluated. The final two steps of this process are the assessment of student and instructor perception regarding the module.

III. Implementation Status in CS/2: CS 2401

In the winter quarter of 2011-12, we introduced a module into the CS/2 course (CS 240B) at Ohio University that covered a simplified RAM model of computing (using the GNU GIMPLE approach), parallel thinking about computing via the PRAM model, and an introduction to parallel algorithms via a sorting example. The module also touched on divide and conquer approaches and algorithm analysis.

The purpose of the module in CS/2 was to introduce students to (i) machine models of computing, (ii) parallelism, and (iii) algorithm analysis for both sequential and parallel computation. The expectation was that students would achieve the knowledge level on Bloom's taxonomy on the material that was covered, and that, likely, that the material would be covered again later in the curriculum.

The performance indicator for this module was an online quiz that tested students knowledge of the presented material. Forty-eight students completed the assessment. Unfortunately, only 50% of the students achieved the desired

[1]Formerly CS 240B/C
[2]Formerly CS 361
[3]Formerly CS 442

result of “excellent” or “good.” The decision was made to repeat this material (and the assessment) in a later course.

IV. Implementation in DS/A

During the spring quarter of 2011-12, a week-long series of lectures (four total hours) were designed for the *Data Structures* course that covered parallel performance, thread creation, thread safety, cache coherence and false-sharing, task scheduling, performance limitations, Amdahl’s law and parallel programming. Two blackboard quizzes were created to tests the students knowledge (K) of the material, and a programming assignment that tested theirability to apply their knowledge. The results of these assessments were measured to see whether the presentation of the material was effective.

Rating	Total	Percent
Excellent ($\geq$ 90%)	15	42
Good ($\geq$ 70%)	18	50
Developing ($\geq$ 50%)	3	8
Poor ($<$ 50 %)	0	0

Table I
Thread Creation (Knowledge) Assessment

Rating	Total	Percent
Excellent ($\geq$ 90%)	14	39
Good ($\geq$ 70%)	15	42
Developing ($\geq$ 50%)	7	19
Poor ($<$ 50 %)	0	0

Table II
Parallel Performance (Knowledge) Assessment

As we can see in tables I and II above, students in this course were able to demonstrate knowledge of the material through relative strong performances in both assessment. Their ability to apply the material (table III below) in the programming assignment was not as successful since less than 60% of the students scored in the excellent or good range. This specific module/exercise will be re-examined during the next iteration of this implementation.

Rating	Total	Percent
Excellent ($\geq$ 90%)	12	37.5
Good ($\geq$ 70%)	7	21.875
Developing ($\geq$ 50%)	8	25
Poor ($<$ 50 %)	5	15.625

Table III
Parallel Performance (Application) Assessment

V. Implementation in CS 4000

Starting in the spring semester of 2012-13, Ohio University began offering a *required* course in parallel and distributed computing: CS 4000 Introduction to Distributed, Parallel, and Web-Centric Computing. This course uses the text *An Introduction to Parallel Programming* [2] along with extensive notes to provide students with a deep and practical introduction to various aspect of parallel and distrubuted computing. The modules and associated performance indicators are currently under development for the course. At present, the course has covered: parallel algorithms (parallel prefix sums), parallel models of computation, parallel and distributed programming (OpenMP, Pthreads, OpenMPI) and related topics.

VI. Implementation Status in CS 4420

During the winter quarter of 2011-12, and fall and spring semesters of 2012-13, the course content in the CS 4420 has been revised and updated to cover material from the curricular recommendations on parallel and distributed computing. The course covers 2.5 hours of architecture topics, 2.75 hours of algorithms topics, 9.5 hours of programming topics, and two hours of cross-cutting topics. In this course, students were given an assignment involving a multi-threaded programming assignment that involved finding bounding boxes of characters in images. In this case, the results were encouraging. 75% achieved a good or excellent result. Unfortunately, the sample size (4) was small for this specific offering of CS 442.

VII. Results

As a unique test case, we repeated the module that covered parallel models of computation in both CS/2 (CS 240B) and CS 4000, and we performed the same assessment. The results of the assessment of performance on the online-quiz is provided in Table IV.

	CS 2		CS 4000	
Rating	Total	Percent	Total	Percent
Excellent ($\geq$ 90%)	2	4	7	36
Good ($\geq$ 70%)	22	46	9	47
Developing ($\geq$ 50%)	18	38	2	11
Poor ($<$ 50 %)	6	13	1	5

Table IV
Comparing Early vs. Late Results on Models of Computation

Clearly, the students at the 4000-level are able to grasp this material on parallel models of computation better than at the 2000 level. This is, perhaps, a good choice for material to be covered in more than one location in the curriculum.

References

[1] S. K. Prasad *et al.* (2010) NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing — core topics for undergraduates — preliminary version. [Online]. Available: http://www.cs.gsu.edu/ tcpp/curriculum/index.php

[2] P. Pacheco, *An Introduction to Parallel Programming*. Morgan Kaufman, 2011.