Incorporating Parallel and Distributed Computing Across a Liberal Arts Computer Science Curriculum

Tia Newhall, Andrew Danner, Kevin Webb
Computer Science Department
Swarthmore College
Swarthmore, PA 19081, USA
{newhall, adanner, kwebb}@cs.swarthmore.edu

Abstract—We discuss changes that we made to incorporate parallel and distributed computing into all levels of our undergraduate computer science curriculum. These changes were part of our 2012 NSF/IEEE-TCPP Early Adopter [1] effort and a response to the growing importance of including coverage of these topics in the core undergraduate CS curriculum [2]. The main goal of our effort is to ensure that every CS graduate has exposure to parallel and distributed computing, with both a breadth and depth of coverage in different contexts. We focus on teaching students “parallel thinking”, exposing them to the fundamental issues of parallel and distributed computing—algorithms, systems, architecture, programming perspectives, and the skills required to analyze and solve problems in a distributed environment.

The center of our changes is a new intermediate-level course that introduces students to computer systems and to parallel computing. This new course is a requirement for the CS major and minor and serves as a prerequisite to many upper-level courses that provide more breadth and depth of coverage of parallel and distributed computing topics. To date we have added and expanded coverage of parallel and distributed computing into seven upper-level courses. Our plan is to continue this expansion both within courses that already have some content and also into courses that have traditionally not have such coverage. We discuss our experiences developing our curriculum and evaluate its implementation.

I. BACKGROUND

Computer Science curricula at liberal arts colleges are often constrained by the small size of the department and by the requirements of a general liberal arts education. For example, at Swarthmore College, students must take 20 of their 32 courses outside their major, and our department is only able to offer each individual upper-level course once every other year. Additionally, our introductory course must serve a diverse set of students—it is both an entry point to the CS major and a general science credit for non-majors.

The result of these constraints is that our curriculum must have a shallow pre-requisite hierarchy and must provide multiple options for satisfying the major. Also, our courses need to be tailored to accommodate a wide variety of student backgrounds. For example, in any upper-level course, there may be advanced senior CS majors along side freshmen taking their very first advanced course.

Prior to our curricular changes, students could graduate with a CS major without ever being exposed to computer systems or to parallel and distributed computing. Our changes now ensure that every graduating CS major has both broad and in-depth exposure to these important topics.

Although our curriculum is specifically designed for a liberal arts college, we feel that many of its ideas could be applied to other types of institutions.

II. CURRICULUM OVERVIEW

In Fall 2012, we began integrating topics from the TCPP curriculum into our courses and modified our CS major and minor requirements to ensure that every student is exposed to distributed computing topics in both introductory and advanced courses. Our approach involved adding a new required intermediate-level course, Introduction to Computer Systems, that provides every student with a background in computer organization, computer systems, and parallel computing, focusing on shared memory parallelism. It serves as a prerequisite to about one half of our upper-level courses, allowing us to remove introductory and background material from those courses to make room for more advanced content.

Additionally, we organized our upper-level courses into three groups (Systems, Applications, and Theory), and require CS majors to take at least one course from each. This grouping ensures that students experience both breadth and depth of coverage for TCPP topics. It also exposes them to these topics in different contexts—from systems, applications, languages, and theoretical perspectives. Figure 1 shows our new curricular structure, with the prerequisite hierarchy including our newly added course, and the grouping of our upper-level courses. The courses listed in dark boxes are the ones to which we added or expanded coverage of TCPP topics.

A. Introduction To Computer Systems

Introduction To Computer Systems was first offered in Fall 2012 and has been offered every semester since then. It is required for all CS majors and minors, and it serves as a first introduction to machine organization and computer systems. We have also made it a new prerequisite to many of our upper-level courses, as it provides students with an appropriate background in systems, assembly, and parallel computing.

The course includes many topics from the TCPP curriculum, with a focus on covering the minimal skill set. Topics covered span Architecture, Programming, and Algorithms from the TCPP curriculum including: the memory hierarchy, disk,
Figs. 1. Course Grouping Structure and Prerequisite Hierarchy including our new Introduction to Computer Systems requirement. Courses in dark blue boxes are the ones to which we added or expanded parallel and distributed content. After taking CS1, students can take CS2 or Introduction to Systems in any order. CS2 is a prerequisite to all upper-level courses. Introduction to Computer Systems is a prerequisite to about one half of upper-level courses. Upper-level courses are divided into three groups (Systems, Applications, and Theory). Majors must take at least one course from each group.

Caching and cache coherence, operating systems, processes, virtual memory, buses, multi-core and SMP, pipelines, implicit parallelism, shared memory programming, threads, synchronization, deadlock, race conditions, critical sections, producer-consumer, Amdahl’s law, scalability, speed-up, and an introduction to distributed computing and message passing.

B. Advanced Courses

To date we have added and expanded parallel and distributed computing topics into seven of our upper-level courses. In four courses we added advanced coverage of TCPP topics where there previously was little to none. These include:

- **Algorithms**: Added Cilk, Work, Span, and Parallel Merge Sort
- **Graphics**: Added SIMD processing, GPU and CUDA programming.
- **Operating Systems**: Added intro to distributed systems and distributed file systems. Expanded coverage of threads, synchronization, message passing and sockets.
- **Databases**: Added parallel join methods, distributed transactions.

In other courses, nearly all of the content is part of the TCPP curriculum:

- **Parallel and Distributed Computing**: Broad and in-depth coverage of parallel and distributed computing topics. Topic coverage is focused roughly as 1/3 languages, 1/3 algorithms, and 1/3 systems topics.
- **Cloud Computing**: Broad and in-depth coverage of large-scale systems and datacenter topics. Includes distributed resource management, replication and consistency, MapReduce, and distributed key-value stores.
- **Networking**: Broad and in-depth coverage of communication networks and their applications. Includes client/server and peer-to-peer communication models, serving concurrent requests, and distributed routing algorithms.

Parallel and Distributed Computing and Cloud Computing are seminar-style courses where students read and discuss primary work in the field. In addition to traditional assignments, students propose and carry out independent, multi-week projects in these courses, thus increasing opportunities for students to participate in parallel and distributed research.

Whether all or part of the curriculum in these courses is TCPP topics, we were able to expand coverage and expand the depth of coverage of these topics in all of these courses by removing introductory topics to make room for more advanced material; with the addition of our new Introduction to Computer Systems course as a prerequisite, we are able to start at a more advanced level in each of these courses.

III. SUMMARY

To date, we are starting the third year of our new curriculum, and we are very pleased with the changes we made. Both our students and our faculty feel that the addition of the Introduction to Computer Systems is the key to giving students the necessary background to prepare them for more advanced course work in the upper-level courses that require it. Additionally, we believe this common introductory course succeeded in achieving the main goal of our effort: that every Swarthmore CS major and minor is exposed to parallel and distributed computing. Grouping upper-level courses into three categories and providing multiple advanced options with TCPP content ensures that every student has the opportunity to further explore fundamental issues in parallel and distributed computing from the algorithmic, systems, architecture, and programming perspectives. In these courses, they master the skills necessary to analyze and problem solve in parallel and distributed environments.

We continue to develop our courses to better cover TCPP content and we look to expand TCPP coverage into more courses. We plan to add TCPP coverage to our Compilers course the next time it is taught, and may include it in some of our other upper-level Applications group courses in the future.

REFERENCES
