TCPP Early Adopter Fall 2012

Tia Newhall and Andrew Danner
Swarthmore College, Department of Computer Science

1 Overview

We propose to integrate elements of the TCPP curriculum into multiple undergraduate courses over multiple semesters. Starting Fall 2012, we are adding a new course required for all CS majors and offered every semester, *CS31: Introduction to Computers and Computer Systems*. The course will include introductory TCPP material and be a prerequisite for upper level courses that build on parallel and distributed topics in different contexts.

Existing upper level electives are offered every other year. We propose to introduce or develop additional TCPP material for *CS41: Algorithms* (starting Fall 2012), *CS40: Graphics* (Spring 2013), *CS45: Operating Systems* (Fall 2013), and *CS87: Parallel and Distributed Computing* (Spring 2014). The proposed curricular changes and course schedule will provide at least one introductory and one upper level course containing TCPP core topics every semester.

Co-PI Newhall will introduce CS31, in addition to modifying and enhancing CS 45 and CS 87. Co-PI Danner will modify and enhance CS 40 and CS 41.

Project Goals

The main goal of our proposed TCPP Early Adopter effort is to ensure that every Swarthmore CS major and minor is exposed to parallel and distributed computing. In particular, we will focus on teaching students "parallel thinking". We want every student to be exposed to fundamental issues in parallel and distributed computing from the algorithmic, systems, architecture, and programming perspectives. Students should also develop skills to analyze and problem solve in parallel and distributed environments.

In addition to our primary goal we also want to increase opportunities for students to participate in parallel and distributed research projects. We hope to share course materials that we develop such as lectures, lab assignments, and syllabi, with colleagues at other CS departments who are interested in adding TCPP content to similar courses.

Swarthmore Computer Science

Swarthmore is a small, elite liberal arts college. The CS Department consists of five tenure track faculty and offers CS major and minor degrees. Although we are small, we try to cover a broad range of computer science through our upper-level course offerings. Our current introductory sequence provides students with background in algorithmic problem solving, machine organization, and data structures and algorithms. These courses are offered every year. Upper-level courses are each typically offered once every other year. A large number of our graduates will eventually go on to CS graduate school. For this reason, our curriculum also includes a focus on preparing students for graduate study by providing them instruction and practice in reading and discussing CS research papers, technical writing, oral presentation, and independent research projects.

High-level Description of our Plan

We plan to achieve the main goal of our proposal by adding much of the TCPP Curriculum into several of our current courses, modifying existing course content to include these topics, and strengthening current coverage in others. Additionally, we will add a new introductory course, CS31, to prepare students for upper level courses featuring distributed and parallel topics.

Our department's curriculum already has an emphasis on algorithmic problem solving. With this initiative we will add an element of parallel thinking to the curriculum as well. We were delighted to see that the ACM-IEEE Computer Science Curricuala CS2013 Strawman Draft report has a new knowledge area devoted to parallel computing and that parallel and distributed computing topics appear in many other knowledge areas. We support the approach of the TCPP Initiative: that exposure to these topics

should not be confined to one particular course but that it should be covered in several courses so that students will have multiple opportunities to see parallel and distributed computing and apply it in multiple contexts.

The co-PIs have already introduced parallel and distributed computing topics into existing courses, and recently developed a course devoted to parallel and distributed computing. We currently expose some of our students to many of the topics in the TCPP Curriculum. However, the effort we propose will expand coverage of these topics and ensure that every graduating CS major and minor gains exposure to parallel and distributed computing.

2 Individual Course Details

CS31: Introduction to Computers and Computer Systems. Our current introductory sequence consists of a standard CS1 course in Python and a CS2 course on data structures and algorithms in C++. These courses provide students with a solid background in algorithmic problem solving and analysis needed for upper-level CS courses. However, our students often have little to no background in computer systems, and CS31 will provide this.

A main learning goal of CS31 is to understand how a program goes from a high-level parallel or sequential programming language to being executed on a computer. The course will be structured much like a vertical slice through the computer, starting from representing bits and digital logic to ending with executing programs written in a high-level language. We will consider both sequential programs written in C and parallel programs running on multicore computers written using pthreads. CS31 will introduce topics in machine organization, architecture, OS, compilers, and parallel programming. It will serve as a new prerequisite to many of our upper-level courses, providing students with appropriate background in systems, assembly, and parallel computing. It will be a new prerequisite for the following upper-level courses: OS, Compilers, Parallel and Distributed Computing, Graphics, Computer Networks, and Database Systems. In addition, it will ensure that every graduating CS major and minor has had some exposure to these important topics. As a secondary goal it will provide students instruction in the C programming language and in using many system and programming tools including gdb, valgrind, make, bash, revision control software, `/proc`, and gprof.

The course will include many topics from from the TCPP curriculum, particularly those from the Architecture Topics, but it will also introduce many from the Programming Topics and a few from the Algorithms Topics. The course will have a strong focus on analyzing problems from a systems perspective, for example using knowledge of the memory hierarchy to evaluate space usage as a factor of performance in addition to using big-O analysis. The course will examine measurement and analysis at various levels including performance metrics, latency, bandwidth, I/O and synchronization costs, Amdahl's Law, space-time tradeoffs, locality, and speedup. CS31 will also provide an introduction to operating systems and parallel programming, focusing on shared memory programming with pthreads.

CS40: Computer Graphics. The primary focus for the Graphics course is on data structures and algorithms for representing and rendering 3D models. We have gradually introduced parallel topics, primarily CUDA, into the course. In two weeks of instruction in Spring 2011 we provided a brief introduction into CUDA and covered TCPP core topics including SIMD and stream architectures, memory organization (CPU memory, GPU memory, shared memory), hybrid computing, GPU threads, synchronization, scheduling on CUDA GPUs, data layout, and speedups. Practical examples and assignments explored both graphics and general purpose applications, including parallel reductions on large arrays. Under the TCPP grant, we plan to further develop these topics, possibly expanding into a third week of coverage. By redesigning some of the introductory material and introducing programmable shaders earlier, students can begin to understand GPU programming well before we begin a full discussion of GPGPU programming using CUDA. This will allow us to better integrate TCPP topics without sacrificing more traditional core material.

CS41: Algorithms. This course explores fundamentals of Algorithmic design and analysis, including formalizing an algorithmic statement of a problem from abstract descriptions, developing algorithmic solutions, proving correctness, and analyzing both runtime and space complexity. While the course traditionally uses the RAM model of computation throughout, co-PI Danner has experience in

both the I/O-model, or out-of-core model, and the PRAM model of computation. The course already covers some TCPP topics including asymptotic analysis, time and space complexity, divide and conquer, and recursive techniques. As part of our grant proposal we plan to introduce two to three weeks of material covering alternative computational models including PRAM and out-of-core. We will use merge sort as a primary example revisiting the analysis of its complexity in the RAM, PRAM, and out-of-core contexts. We will add core TCPP concepts including speedup, scalability, work, and span in the discussion of our analysis. Additionally, we intend to introduce other examples of parallel computation including reductions on large arrays.

A perhaps uncommon feature of our Algorithms course is a recently added lab component. As part of the TCPP grant, we will create at least one lab model which explores some parallel algorithm in practice. Some care is needed to design this exercise appropriately, as students will only have sequential programming courses as prerequisites and may not be familiar with practical parallel programming tools. We propose to design a basic wrapper around either CUDA or Intel's TBB to provide a simple interface to exploring basic parallel algorithmic principles in practice.

CS45: Operating Systems. CS45 covers a fairly standard undergraduate OS curriculum, including processes, threads, synchronization, memory management, file systems, I/O, protection and security and an introduction to distributed systems. The course strives to have a good balance of theory and practice. It includes a strong focus on analyzing performance based on systems costs, trade-offs in system design, layered design, and the separation of mechanism and policy.

Prior to CS31 being added to our curriculum, this course has also provide and introduction to C programming and C programming tools, computer architecture, and Unix utilities. With the addition of CS31 as a new prerequisite to CS45, we will be able to cover more advanced operating systems topics in OS. In particular to TCPP, we will add more coverage of distributed systems, distributed file systems, networking and security. Because students will have some experience with pthread programming in CS31, there will also be an opportunity to have students implement and test some of the complicated synchronization problems that they solve in OS, but have only evaluated as written problems in the past.

CS87: Parallel and Distributed Computing. We first added CS87 to our curriculum in Spring 2010, and offered it a second time in Spring 2012. This course is a broad survey of parallel and distributed computing topics. CS87 includes both lecture and in-class discussion of CS research papers. The course includes many short lab assignments in the first half of the semester to give students practice using different parallel and distributed programing languages and paradigms. These have included pthreads, MPI, OpenMP, C socket client-server, CUDA, and practice using XSEDE resources for MPI and hybrid MPI-CUDA programming. The short labs are designed to give tools for carrying out independent course projects during the second half of the course. The course is about 1/3 systems topics, 1/3 programming languages, and 1/3 algorithms. There is an emphasis on paper reading, writing, oral presentation, experimentation, and researching, proposing, and carrying out an independent project.

The course covers many of the TCPP topics in all four Areas. Topics covered in the most recent offering include: the memory hierarchy, pipelining, multicore, SMPs, false sharing, vector processors, GPUs, MPPs, clusters, grid, P2P, cloud computing, SIMD, MIMD, data parallel, client-server, distributed memory, shared memory, threads, synchronization, MPI, CUDA, OpenMP, Map-Reduce, hybrid CPU-GPU-MPI programming, messaging communication, parallel programming patterns, parallel reduce and scan, trade-offs, speed-up, scalability, dependencies, time, power, parallel algorithms, fault tolerance, DFS, DSM, security, and networking.

Because students have a wide range of systems background entering this course, we have had to provide instruction in basic systems, architecture and C programming. With the addition of CS31 as a new prerequisite, we will be able to assume that all students have background in architecture, systems, C programming, some parallel and distributed computing topics, as well as practice with a systems perspective to performance analysis. With this background much of the introductory material in CS87 can be replaced with more advanced parallel and distributed computing topics. This will allow for increasing both the breadth and depth of coverage of these two fields. It will also allow for at least one additional parallel or distributed short lab to replace a C warm-up lab that

was necessary prior to CS31 as a prerequisite. Most likely the additional lab will involve using Hadoop.

3 Project Evaluation

Our primary evaluation goals for this project are to assess how well we integrate TCPP topics across the curriculum and how our modifications influence student ability to think effectively using parallel concepts. Immediate assessment of individual course will be done through lab assignments, exams, and end of course surveys. However, we are particularly interested in how topics introduced in CS31 prepare students for upper level courses that explore advanced parallel topics. Naturally, as we introduce topics across multiple courses and phase in the prerequisite of CS31, there will be some redundancy and repetition across upper level course. We plan to evaluate and identify these common topics and consider them for inclusion into later iterations of CS31.

Ideally, we would like students to apply knowledge of parallel topics to other courses in the curriculum which do not necessarily emphasize TCPP core topics. Spring 2013 presents an opportunity to evaluate how topics in CS31 and CS41 are retained and applied as there will likely be some students in CS40 which have had one or both of these courses in the previous semester. Since CS31 will be a prerequisite for CS40, but not for CS41, we plan to assess the level of overlap in topic coverage amongst these courses and potentially adjust the coverage in future iterations of these courses.

As we coalesce common TCPP core topics from upper level courses in CS31, there may be opportunities to move smaller, simpler elements of parallel programming into other introductory level courses including our CS1 and CS2 equivalent courses, that currently do not emphasize parallel concepts.

4 Budget

We currently have sufficient hardware resources to support parallel computing in our teaching labs, including local and remote computing clusters. Our primary support need is for course development. As both co-PIs will undertake significant modifications of the departmental curriculum across multiple courses and multiple semesters, we are requesting a budget $2500 for faculty support of course development and evaluation. Part of this support may also be used to supplement the travel budget of the co-PIs so they may attend workshops or conferences to exchange ideas and summarize their experience in implementing TCPP core topics in an undergraduate curriculum.

5 Faculty Bios

Tia Newhall is an Associate Professor of Computer Science and Swarthmore College. She joined the department in 1999. Tia conducts research in parallel and distributed systems. Her main research project, Nswap, address problems related to providing fast backing storage in cluster systems to better support data intensive parallel computing. In addition to Nswap, Tia has conducted research in parallel tools, parallel programming languages, and OS support for SMPs. Tia regularly teaches the majorrity of our systems courses including Operating Systems, Parallel and Distributed Computing, Compilers, and Database Management Systems, in addition to teaching introductory courses. Tia will incorporate and strengthen TCPP content in three courses: OS, Parallel and Distributed Computing, and Introduction to Computer Systems.

Andrew Danner is an Assistant Professor of Computer Science at Swarthmore College, which he joined in 2006. His research interests are in the area of I/O-efficient algorithms, also called out-of-core or external-memory algorithms. Andrew applies I/O-efficient algorithms to applications in Geographic Information Systems (GIS), particularly terrain modeling. Recently, he has incorporated MPI and GPGPU solutions using CUDA to further accelerate the processing of very large data sets and he has worked with students over the summer to explore these research questions in parallel computing. Andrew regularly teaches Algorithms, Theory of Computation, and Graphics, in addition to introductory topics in CS.