Integrating the NSF/TCPP Curriculum Guidelines into Undergraduate Systems Courses at Portland State University

Karen L. Karavanic, Associate Professor of Computer Science

I. Introduction

We propose to integrate portions of the NSF/TCPP Guidelines into a series of systems courses in the undergraduate curriculum at Portland State University. The core (required) courses included in this proposal are: CS201 Computer Systems Programming; and CS 333 Introduction to Operating Systems. The proposal also includes CS 4xx Introduction to Performance Measurement, Modeling and Analysis, a new elective course to be developed as part of this proposal that we hope to add as a permanent course. CS 201 and CS 333 are required one-quarter courses in the CS undergraduate curriculum. We are designing our changes as an integrated set designed to carry forward through these courses. CS 201 is a prerequisite for CS 333, and both would be prerequisites for CS 4xx.

Our CS major population includes two groups that we specifically address with additional components: a relatively large percentage of transfer students from Portland Community College (approx. 30% of CS majors) who would have taken CS 201 *before* entry into our Department; and post-baccalaureates ("post-bacs") who take selective courses as preparation to enter the Masters CS program with a non-CS Bachelor's Degree. These students would enter our systems track mid-stream. For this reason, we will develop specific video and lab materials designed as self-study units, as prerequisite to the new parts of CS 333 and CS 4xx. This would serve as review for students continuing through our own program; and catch up for those entering midstream. They will also be a mechanism for testing student knowledge at each step in the sequence. Once our own development and testing is complete, we plan to make these materials available to other institutions, in particular to PCC through our well-established communication channels. Expanding the curriculum changes to PCC would greatly increase the reach of our efforts and the number of students reaping the benefits.

II. CS201 Systems Programming

This course has replaced the previous Computer Organization course and uses the Carnegie Mellon text. Students learn the basics of the computer hardware, floating point representation, and the x86 instruction set. The programming projects focus on C programming, that is also taught as part of the course material.

The new modules we develop will focus on three aspects: a measurement-oriented introduction to performance; an introduction to the memory hierarchy; and a comparative architecture module incorporating CUDA and/or SSE instructions (see Table 1). We will develop a lecture and programming exercise that illustrate the GPU hardware, including memory layout on the GPU. The focus is not on the

programming aspects *per se*; however we will provide a brief introduction to programming in CUDA as needed to develop the lesson.

Table 1: Sketch of Module Topics Per Course

Course	Topic	Focus	Programming Ex.?
CS 201	Floating Point Representation	Accuracy	Y
	Memory Hierarchy	Introduction	
	Performance 101	Measurement Tools	Y
	GPU/SSE hardware	Instructions	
		Speedup	Y
CS 333	Memory Hierarchy	Multicore Memory Architectures	
	Multithreaded (shared memory) programming	Comparative Performance of sequential vs. threaded code	Y
		Debugging	Y
		Comparison to Simultaneous Multi-Threading	
	Multicore Scheduling	Scheduling Algorithms	Y
		Power-Aware Scheduling	
CS 4xx	Metrics	CPU Utilization	Y
	I/O	Bandwidth & Contention	Y
	Power	FLOPS vs. FLOPS/Watt	
	High End	TOP500, Green500, Graph500	
	Amdahl's Law	Amdahl's Law in the Multicore Era	
	Hybrid Accelerated Computing	GPU vs. CPU Performance	Y
	MapReduce	Metrics & Measuring Performance	Y

III. CS333 Introduction to Operating Systems

Over the past two academic years I have initiated and implemented an alternative structure for this course, that adds a mandatory, closed lab programming

component. "Closed lab" refers to the setup wherein students register for a 3-hour weekly laboratory session in addition to the lecture. Each lab section is limited to under 16 students. Students are each seated at a Linux workstation that is disconnected from the internet and email, and login to a special directory for their lab work only. Each week they complete one multi-part programming exercise in the lab, with a lab instructor present to guide their work. This approach has been previously studied by others in introductory courses; we extended the idea to the junior-level OS course. Our initial evaluation demonstrates that this method accomplishes the key educational goals (minimum programming competency across all students, an accelerated learning process that provides fast feedback and course corrections, and a positive environment that promotes confidence by allowing students to see their own work in relation to others). We have already tested out two exercises that use pthreads. We will develop and test two additional lab units based on the TCPP guidelines.

The new module we develop will include lecture notes plus programming exercises based on topics included in the "Architecture" and "Programming" areas. In particular, the memory hierarchy and multithreading topics from Architecture; and the distributed memory / threading topics of Programming. One laboratory will include the use of tools for debugging and performance analysis. We will also add new material on scheduling to emphasize multicore scheduling and to introduce the concept as power as a first-class resource.

IV. CS 4xx Introduction to Performance Measurement, Modeling, and Analysis

Although there are frequent performance-related offerings available to undergraduates, we plan to streamline these by the addition of a permanent single-quarter course to serve as an introduction to performance. This course would introduce metrics, measurement techniques, modeling, simulation, and data analysis. Prerequisites would include CS 333 and also the required introduction to statistics course. An example text would be Lilja's Performance Evaluation.

V. Evaluation Plan

We will directly evaluate our efforts through student "before and after" tests in each of the modified courses. We will test for the particular elements listed in Table 1. In addition, we will conduct cohort analysis by tracking the students from the modified CS 201 course through CS 333 and the new elective CS 4xx. We regularly offer 3-4 sections each of CS 201 and CS 333 each year, opening the possibility to do comparative evaluation by testing all students, those who have and who have not completed the new modules. Our evaluation has several different goals, including evaluation of the flow of the concepts through the courses as we hope to build on the foundational concepts of concurrency.

VI. Budget Sketch

We request the maximum budget of $2500. We will use the funds for: hourly pay for a student assistant; facilities costs for preparing the lesson module video's; and PI travel to relevant conferences or meetings. We plan to hire a student hourly worker for programming time at the CS Department usual rate (currently $12.50 / hour). The student worker will develop and test the programming exercises and homeworks. The student will be either an undergraduate (most likely) or masters level student. We plan to develop materials for reuse, including video and written materials, for students entering our program after the CS 201 course. These will also be disseminated more widely. Therefore we budget for any facilities costs for recording and editing. Finally, we budget for travel to relevant conferences or meetings for the project PI.

VII. Faculty Bio

Karen Karavanic is an Associate Professor of Computer Science at Portland State University. Her research is in tools for performance measurement and analysis of high end applications. She regularly teaches the required undergraduate and graduate courses in Operating Systems, and has also taught the required undergraduate course in Systems Programming. Her particular educational focus is on integrating research into the classroom. Since joining the faculty at Portland State she has developed and offered a series of mixed undergraduate/graduate courses related to her research areas, including Performance of Heterogeneous Systems, Multicore Computing, and General Purpose GPU Computing; students in these courses complete group research projects. With collaborators Barry Wilkinson, Jens Mache, and David Bunde, she presented a full-day session "An Educator's Toolbox for CUDA" at SC'12 as part of the HPC Educators Program.