High-Performance Computing in High Schools?

Tom Murphy • Contra Costa College

In a previous installment of this column (April 2006, http://csdl2.computer.org/persagen/DLABsToc.jsp?resourcePath=/dl/mags/ds/&toc=comp/mags/ds/2006/04/o4toc.xml), I wrote about implementing high-performance computing at Contra Costa College, a community college near Berkeley. We established a two-year vocational-training program for future Linux cluster administrators. Unfortunately, CCC has temporarily suspended its HPC degree program owing to a lack of students. The dot-com bust caused a downturn in all high-tech fields, particularly in Northern California. Further complicating matters is the lack of a consistent job title for cluster administrators, which makes it hard to point people to available jobs. Consequently, students don’t know how bright a future the HPC job field has, and only a few students seek HPC training.

However, all is not lost; we’ve salvaged the bulk of the HPC classes by making them part of our Computer Information Systems program. The remaining capstone HPC courses will lie dormant in the catalog until we can reopen the HPC AS degree. Furthermore, our HPC computational cluster hardware and related experience with computational models have inspired and enabled the beginnings of a Computational Science and Engineering degree program at CCC.

HPC brought to high school students

The planned Computational Science and Engineering major will include core mathematics, computer science, biology, chemistry, and physics courses along with Introduction to Computational Science and a computational-modeling course for good measure. This is a daunting sequence of courses for the average two-year community college student. The HPC technician program taught us the importance of developing an adequate supply of students. The prospect of having only a few average community college students has led us to train a cadre of uncommonly bright high school students recruited from my partner high school, Middle College High School. This gives me four to six years to foster amazing Computational Science and Engineering graduates.

MCHS is part of a nationwide middle-college movement, where a community college partners with a high school to augment the high school curriculum with additional courses. Ours is unique among such partnerships because there aren't separate high school facilities. MCHS holds its high school classes in our college classrooms. Furthermore, it offers no advanced-placement science classes and only a few high school science classes. Instead, most of the science classes are actually CCC science classes, with the high school students registered as college students via concurrent enrollment. The end result is that if I can convince the high school students to take a specific series of courses, I can have a new group of high school seniors who have already taken most of the science courses the Computational Science and Engineering major requires.

Helping me convince the students to take courses is our three-year Tech Prep grant. Tech Prep is funded through the Carl D. Perkins Vocational and Technical Education Act of 1998. It’s a national program that gives high schools grants to foster 2 + 2 programs, where high school juniors and seniors take specialized courses that guide them to a two-year AA degree. My work with the Tech Prep grant involved creating an Applied Computer Science certificate primarily awarded to seniors. With only another year of study, they can earn an AS in computer science. When the Computational Science and Engineering AS degree is official, I’ll add an Applied Computational Science track and certificate for MCHS students as well.
Robotics and HPC

My high school HPC program starts by reaching out to high school freshman via an open-entry, open-exit robotics program. "A robotics program," you say, "has nothing to do with HPC!" And at first glance, you’re right. However, creating a robot involves physical modeling with similar constraints, reasoning, and compromises as computational modeling. The programming skills necessary to coordinate time-sensitive parallel robotic tasks are similar to those necessary to coordinate time-sensitive MPI-based computational tasks. Like Mr. Miyagi teaching karate fundamentals through fence painting in *The Karate Kid*, robotics provides an open-ended environment that captures students’ imagination and interest while guiding them down creative scientific paths.

Student projects

MCHS seniors must have a project as part of their graduation program, so I used this requirement to support a COMP 42 Special Projects in Applied Computer Science course. The course’s topics range over life, the universe, and everything—provided I approve the student’s particular topic. It’s an open-entry, open-exit lab course, potentially spanning multiple semesters or even years. We have great computational resources supporting the class: CCC has four clusters. We also have outside support available from the Oklahoma University Supercomputing Center for Education & Research (http://www.oscer.ou.edu/), the San Diego Supercomputing Center (http://www.sdsc.edu/), and the National Center for Supercomputing Applications (http://www.ncsa.uiuc.edu/). Animation projects are a particularly fruitful area to use our clusters, called render farms in the animation context.

A wide variety of projects are already underway under the COMP 42 umbrella. One involves searching both the Computational Science and Engineering Reference Desk (CSERD) and the National Science Digital Library (NSDL) for one-hour, open-ended explorations, such as predator-prey models, modeling probability with forest-fire simulations, and infinite series with help from Zeno's paradox designing and building a cluster in a suitcase, a LISP-based robot, and computer video games. For years, our students have said they were interested in computer science and, in the next breath, that they wanted to create computer games. So, I’ll now grant their request. Next semester, we’ll explore computer gaming via the Torque Game Engine 3D, which undoubtedly will show students the benefit of additional math and computer science courses.

Introduction to Computational Science

The center and academic foundation of the high school HPC program is the Introduction to Computational Science course, which we’ll offer for the first time in August 2007. I’ll use *Introduction to Computational Science: Modeling and Simulation for the Sciences* (Angela Shiflet, Princeton Univ. Press, 2006), which focuses on system dynamics models (with their global views of systems that change with time) and on cellular-automaton simulations (with their local views of how individuals affect individuals). We’ll also use NetLogo, a cross-platform multiagent programmable modeling environment, which supports both system dynamics and cellular-automaton modeling. We’ll also incorporate material from CSERD and NSDL as well as real-world data sets. The course targets students with limited mathematical sophistication (that is, it requires only a minimum of high school algebra). Luckily, many students will be enrolled concurrently in first-semester calculus.

The course begins with an overview fostering the importance of a computational science that enables *in silico* scientific investigation, verification, and validation. We’ll use NetLogo to investigate uncontrolled and controlled growth by modeling a flock of sheep. Students will explore the factors necessary to model and the notion of what’s “good enough” in crafting a model. We’ll then repeat this process with the spread of infection by using US National Institutes of Health epidemic data, whose predictive power we’ll validate using data from a second and third epidemic. We’ll also examine academic papers describing more sophisticated models and adjustments made tuning the models to better reflect actual history.

We’ll investigate the science used and the issues surrounding all the models; the trade-offs and constraints used in creating the models; and their strengths, weaknesses, and predictive power. The goal is to provide a computational modeling framework for the students to use as a guide as they proceed through their other science courses.
This study won’t advance the body of knowledge in either animal husbandry or epidemiology, but it will accomplish three goals:

- Establish that a simple model that broadly describes phenomena can still be strongly predictive.
- Illustrate the ease of constructing such models.
- Develop the scientific intuition that later powers the student through requisite math and science courses, enabling them to contribute to the body of knowledge.

The next issue to investigate is the notion of “good enough” in regard to errors in the model. I’ll also touch on chaos theory, which grew in part out of the experience of “acceptable strange errors.” This will lead into a brief foray into some underlying math—not for mastery, but for a positive first exposure. We’ll look at Euler’s method and Newton’s method and their origins. We’ll also derive the basics of integral and differential calculus. Other subjects we’ll examine include Gaussian elimination and the Runge-Kutta method. Then we’ll examine empiricism and heuristics, returning to the notion of “good enough.” Finally, we’ll return to our models to look under the hood at the mathematics NetLogo generated for us, stressing the importance of ultimately grasping that mathematics to keep true ownership and awareness of errors in models.

The next section of the course will consider chaos, randomness, and pseudorandomness. We’ll use these tools to calculate a circle’s area and then discuss why this isn’t the best way to do so. Next, we’ll discuss where Monte Carlo methods excel: bringing Nemo to life via ray tracing, genetic algorithms, cellular automaton modeling, and particle physics. This will lead into cellular automata, where we’ll remodel the same problems we tackled with system dynamics modeling. We’ll also model the spread of a forest fire, discovering the model’s instabilities at certain propagation rates.

The course’s final section will focus on high-performance computing and the joys and challenges of modeling in the many-core world with the Message Passing Interface. We’ll also analyze a parallel solver for a chessboard Sudoku (64 × 64 squares).

This high school HPC program will result in a cadre of community college graduates with multiple science degrees, savvy to computational methods and well positioned for a four-year degree in any scientific discipline—especially for computational science.

Tom Murphy is the computer science program chair at Contra Costa College and director of its Center for High Performance Computing. He’s also the West County robotics coach. Contact him at tmurphy@contracosta.edu; http://contracosta.edu/cs/murphy.

Related Links

- "Education: High-Performance Computing in Community Colleges?" IEEE DS Online (http://doi.ieeecomputersociety.org/10.1109/MDSO.2006.26 )
- "Education: Integrating Parallel and Distributed Computing in Computer Science Curricula," IEEE DS Online (http://doi.ieeecomputersociety.org/10.1109/MDSO.2006.9 )
- IEEE DS Online's Games and Simulation community (http://dsonline.computer.org/portal/site/dsonline/)

Cite this article:
Tom Murphy, "High-Performance Computing in High Schools?" IEEE Distributed Systems Online, vol. 8, no. 8, 2007, art. no. 0708-o8002.