To appear in the Proceedings of Frontiers in Education '96

Curriculum in High-Performance Scientific Computing

E.R. Jessup
Department of Computer Science
University of Colorado
Boulder, Colorado 80309-0430
email: jessup@cs.colorado.edu
URL: http://www.cs.colorado.edu/ jessup/Home.html

The purpose of this paper is to report on an undergraduate course in the use of high-performance computing systems in scientific and engineering applications. The course and the instructional materials on which it is based were developed at the University of Colorado at Boulder (UCB) by Lloyd Fosdick, Elizabeth Jessup, and Carolyn Schauble, all of UCB, and Gitta Domik, of the University of Paderborn. Its development was supported by the National Science Foundation under a CISE Educational Infrastructure grant, and it has been taught at UCB since 1991. The instructional materials are publically available, and their various sources are listed at the end of this paper.

To turn students into effective users of supercomputers, it is essential to introduce them to the machines through real application problems and not just toy problems or simple numerical examples. The students must learn which architectures are appropriate for which problems and how best to map the problems onto those machines. They must also learn how to determine the performance of their programs, including how to interpret what the compiler has done and how to use the clock properly. Finally, the students must learn to use the color, perspective, and animation capabilities of suitable visualization tools to display and interpret the large amounts of data typically generated by supercomputer programs.

The UCB course exposes students to all of these aspects of high-performance scientific computing (HPSC). The course materials include a series of single-topic tutorials, a laboratory manual and accompanying code, short reference guides to subjects such as IEEE arithmetic and some Unix tools, and manuals for the machines and computational tools. These materials and the course itself are designed for upper level undergraduate students. The prerequisites are one semester of numerical computation at the undergraduate level and proficiency in Fortran or C programming.

The course begins with an introduction to four types of high-performance computer architecure: fast workstations with pipelining capabilities, vector computers, and MIMD and SIMD distributed-memory parallel computers. Students study the basics of architecture, programming techniques, and performance measurement. All programming is done on real machines as opposed to simulators. More specifically, students in the course at UCB use DEC, HP, and SGI workstations and an iPSC/2 hypercube multiprocessor located on the UCB campus as well as a MasPar MP-1 (at MasPar) and a Cray Y-MP (at NCAR). Students access the latter two machines via the internet, and the time on those machines has been donated by the owners of the machines. Students thus experience the real difficulties that can be imposed by remote network access, long job queues, or experimental hardware and compilers.

To support their early work, students read review materials on Unix, Make, and networking basics along with tutorials on the four architectures, computer performance, and scientific visualization. They also study the computational and scientific visualization tools Matlab and AVS. The concepts and tools learned in the first few weeks are applied to the solution of numerical problems in molecular dynamics, advection, and tomography. In each case, students study the science behind the application problem, mathematical models of the problem, and the computational methods used to implement those models. They use numerical computation and scientific visualization to study the physical system. They examine the accuracy of the numerical algorithms, and they study the performance of those algorithms on the different architectures.

Of the three applications studied, molecular dynamics is the most simplified. A tutorial covers both the physics of some simple molecular dynamics problems and computational methods for solving them. In particular, it begins with a review of three models of molecular interaction: the Hooke's law model, the Lennard-Jones model, and the hard-sphere model. It proceeds with the derivation of the equations of motion for molecules interacting in one or two dimensions. The next portion of the tutorial concerns numerical methods for solving the equations of motion. The model of molecules connected together as a one-dimension chain is used as a simple model system throughout the tutorial. The equations of motion for this system admit an exact solution, and the final section of the tutorial is devoted to setting up and solving the appropriate eigenvalue problem to estimate the solution numerically. In working through the five lab chapters accompanying the molecular dynamics tutorial, students advance the equations of motion for the one-dimensional chain of molecules numerically and visualize the resulting motion of the chain. They complete this exercise using Euler's method and Verlet's method and compare their results to those obtained via an "exact" solution of the eigenvalue problem. By working through these lab chapters, students learn not only how equations translate into physical motion but also how inexact numerical methods and arithmetic impact their results. In the remaining chapters, students study how these and more complex models of molecular dynamics map to the Cray Y-MP, the Intel iPSC/2, and the MasPar MP-1. The units on advection and tomography follow a similar outline but with a somewhat more advanced formulation of the physics and mathematics involved.

At UCB, the course is taught as two 50-minute lectures and one three-hour supervised lab period per week. Students work through about one lab manual chapter per week and complete term projects. In the early labs, students modify existing codes in order to learn the machine basics and study machine performance. In later labs and projects, they write original programs. There are two projects in the first semester and one in the second. For the first midterm project, students serve as consultants, identifying the computer that best fits their (fictitious) client's needs. For the first semester's final project, students generalize their work on a one-dimensional chain of molecules to study the dynamics of a two-dimensional mesh of molecules. When enough molecules are included, this two-dimensional mesh behaves like a drumhead, so student projects have included audio as well as video displays. For the second semester's final project, students port a familiar code (from the molecular dynamics unit) to a new supercomputer and study and explain the performance of the ported code.

The purpose of this course is to teach students to be knowledgable users of supercomputers as they are used in the real world. The course is therefore intended to attract students from a variety of disciplines. To date, the students have included juniors, seniors, and first year graduate students majoring in computer science, physics, applied mathematics, and aerospace, chemical, electrical, and mechanical engineering. The student evaluations have been extremely positive, and several of our graduates have gone on to related jobs in industry or at university and national laboratories (including NCAR, NOAA, Sandia, and MIT) or into excellent graduate programs. Courses based on our materials are being offered at many other institutions, including both research universities and smaller 4-year colleges.

Although the course has been very well-received, it is not without its problems. The extremely tight curriculum at UCB (particularly in the engineering disciplines) makes it difficult for students to find time for the HPSC elective. In response to this problem, we reorganized the course so that the first semester is a stand-alone course. Even with this change, our own local offering of the course has attracted a fairly small number of students (6-24 in the first semester course, 2-5 in the second semester). We hope, though, that time will bring us larger enrollments. It's a fun course to teach! Other (non-engineering) schools offering the course do report better success with class size.

In addition, the "high-performance" orientation of the course has lead to some problems at smaller schools with fewer resources. The instructors at those schools have been able to get supercomputer time at remote sites, and they report that the materials are accessible to their students. However, they have run into trouble with a lack of fast internet access and the lack of funding needed to purchase and support workstations and visualization tools (especially the AVS program used in our course for 3D visualization.) The increasing availability of freeware and PC or Mac visualization software has helped to lessen the latter problem.

Finally, as in any course involving advanced architectures, keeping the course materials up to date is very difficult and time-consuming. As a result, the list of machines and problems we cover has not changed since the inception of the course. The impact of the aging of the materials varies. For example, the iPSC/2 has gradually become a relic no longer used in research settings. We have not considered this obsolescence a problem so far as it is easy to learn to program other, newer distributed-memory MIMD computers after learning to use the iPSC/2. The greater problem stems from the recent and severe downturn in the supercomputing industry: only a few distributed-memory (MIMD or SIMD) computers are still in production. That, combined with the lower cost and greater availability of workstations, means that more and more parallel computing is actually carried out on networks of workstations. We have not yet added units on such networks or the tools needed to program them. While the application problems are still timely, adding problems from disciplines other than physics would broaden the appeal of the course. We consider the updating and augmentation of our materials to be a high priority matter and are seeking the funding that would allow us to do so.

Course materials (including syllabi and project assignments) are available by anonymous FTP from cs.colorado.edu in the directory /pub/cs/HPSC or via the WWW at http://www.cs.colorado.edu/courses/materials.html. In addition, the tutorials, short references,and tool manuals has been published as the textbook L. Fosdick, E. Jessup, C. Schauble, and G. Domik, *An Introduction to High-Performance Scientific Computing*, MIT Press, 1996.