HPC Undergraduate Curriculum Development at SDSU using SDSC Resources

Education/Technical Paper (#704) for SC'95
Final revision August 15, 1995

Kris Stewart, Associate Professor
Mathematical Sciences Department
San Diego State Univeristy
and
Computational Science Curriculum Coordinator
San Diego Supercomputer Center
(stewart@cs.sdsu.edu)

Keywords

Undergraduate Curriculum Development, HPC, High Performance Computing, Education, HPC Education

Abstract

Results from the development and teaching of a senior-level undergraduate multidisciplinary course in high performance computing are presented. Having been taught four times, there are several "Lesson Learned" presented in this paper. Help from the technical staff at the San Diego Supercomputer Center and support from the National Science Foundation has been instrumental in the evolution of this course. The work of faculty at other universities has influenced the author's courses and is gratefully acknowledged. A subsequent sophomore level course was developed at SDSU and has become part of a voluntary, cooperative program, Undergraduate Computational Science and Engineering.

Introduction

San Diego State University (**SDSU**) is a member of the San Diego Supercomputer Center (**SDSC**) Consortium. This has provided modest amounts of time on the succession of machines available at the Center, from Cray X/MP to Cray Y/MP to the current Cray C90. The author has also been able to attend workshops at SDSC on a regular basis. Access to this facility and its technical staff has been instrumental for the development of the high performance computing course, **CS 575 Supercomputing for the Sciences and Engineering**.

CS 575 is a senior-level, undergraduate course at SDSU. The course was taught twice in 1991 and 1992 as an *experimental course*, then received the campus approvals to be listed in the 1992/93 Undergraduate Catalog and has been taught nearly every spring semester since then. There have been several follow-on activites that the author has participated in directly related to the development and teaching of this course which are summarized in this paper:

High Peformance Computing Curriculum
: gives an overview of the course focussing on the developmental lectures, computational assignments, writing assignments and skills developed by students enrolled in the course

Lessons Learned
: discusses how the course has evolved over the years of teaching it and the author's rationale for the changes

Evolving Curriculum at SDSU
: discusses another course developed at SDSU to introduce students to computational programming. The need for this course became apparent from teaching CS 575.

Cooperative Programs
: lists and briefly describes several Web oriented programs to assist teachers in developing and sharing curricula in High Performance Computing

Acknowledgements

References

High Peformance Computing Curriculum

Catalog Description:
CS 575 - Supercomputing for the Sciences
Interdisciplinary course intended for all science and engineering majors.
o Advanced computing techniques developed for supercomputers
o Overview of architecture, software tools, scientific computing and communications
o Hands-on experience using supercomputers
Prereq.: Extensive programming background in Fortran or C.

Overview Computational science methods and techniques are presented to students in the CS 575 course through lecture and enhanced through their computing projects. Each student selects a science based problem that is of interest to them, tailoring it to their particular academic background. The students become familiar with the computational characteristics of their program by running it on the educational mainframe machine at SDSU, a SUN Sparc 10. The student writes a formal report describing the problem, the performance of their code and the implications of their results on the "science" of their problem. Once students are familiar with their own science projects, they port the code to the Cray, extend the problem and write a final report on the "science" as well as the "performance" of their project on the Cray compared to Sun Sparc 10

The two most dramatic changes incorporated in the Spring 1995 course were:

New textbook: **High Performance Computing** by Kevin Dowd.

More information on this excellent paperback book is given by:
Local Link to HPC text
http://gnn.com/gnn/bus/ora/item/hpc.html (Web Link)

Extensive use of the WWW and graphical Web Browsers
Course handouts distributed via Web cs575.htm (local link)
http://www.stewart.cs.sdsu.edu (Web link)
SDSC makes all documentation (including the Cray User Guide) available via the WWW
http://www.sdsc.edu/Services/Consult/Cray/CUG/CUG_intro.html (Web link)

so it has became essential to incorporate more network communications training for students in the course.

Electronic mail had always been used as the mechanism for students to ask questions outside of office hours. Electronic mail was used to distribute handouts from the instructor. It was the preferred mechanism for students to submit their computational experiments along with accompanying reports. In Fall 1994, SDSU's computer center established two general labs of Xterminals (capable of Xwindow display). For the first time on the SDSU campus, this gives undergraduates access to the graphic interface to the Internet.

1. Cray architecture

The intent is to gain an appreciation for High Performance Computing and to use the Cray C90 as a sample platform. Students come to understand the subtleties of the C90's design from a programmers point of view with the goal to understand "why" the Cray C90 is fast and how to quantify this. Most of these concepts transfer easily to other HPC platforms, even massively parallel platforms since one of the first steps in parallelizing a code is to find the vectorizable loops.

Another excellent source is available from Cray Research Inc. **TR-OPT 1.0 (E) CF77 and Cray Standard C Optimization for Parallel-Vector Systems**. An excellent, Cray-specific training report covering the Fortran and C compilers. This document provides code examples, diagrams, and explanations of crucial vectorization topics (and the conditions that inhibit the compiler), memory organization, performance tools, common optimization techniques, and much more. More information is available online for Cray from http://www.cray.com, WWW link to Cray Research Inc.

2. Communications

The course covers the resources available through the Internet, including:

- Communicating with peers and instructor (via e-mail)
- Using FTP to transfer programs between machines (a crude look at heterogeneous computing)
- Using NCSA Mosaic or Netscape as Graphical Interfaces to the Wide World Web. Using lynx as a text based interface to WWW (over slow phone lines or when without high performance graphics display devise).

3. Monitoring CPU usage

At SDSU, accounts issued to students in typical computer courses are limited only by the amount of allowed file space. When students run programs on the Cray, they will have a fixed amount of CPU time to work with and monitoring a finite resource such as this is a new skill to develop. To aid with this, students are given a first programming project to run on a mainframe at SDSU and introduce the concept of accounting using crude UNIX timing tools (e.g., dtime). During the semester, students are reminded that they must complete their course projects on the Cray without exceeding their CPU allocations. The more sophisticated timing and resource monitoring tools on the Cray can be used to ensure that the student projects do not use up more time than they are allocated for the semester (which is typically 5 to 10 minutes of Cray CPU time for each student).

4. Computer ethics/responsibility

Students in the course choose between four computer-related scenarios and write a one-page essay discussing the behaviors of the individuals involved. The essays are graded on a pass/fail basis (only essays that show a lack of thought or effort "fail"). This is followed by a week long lecture and discussion in class of computer responsibility and the changing nature of the information intensive world we now live in. The goal is to start a dialog between the instructor and students and for the instructor to gain a better understanding of the students' attitudes concerning computer-related issues.

5. Writing

Writing is a key component to this Computer Science course. Most of the students have no problem with the computer ethics essay described above, but they typically have had little experience in writing a technical report. Therefore, an initial computational experiment (problem statement given in the next section) is assigned that consists of the following components:

1. Take a code given by the instructor (in Fortran or C) and modify it to include calls to the UNIX timer, *dtime*
2. Examine the output from your code to gain an understanding of the measured performance.
3. Write a report giving and justifying your conclusion on performance.

This is followed by the main computational experiment which is chosen individually by each student depending on their interests and consists of:

4. Solve a science-oreinted problem on a campus mainframe and document the mainframe's performance in a written report.
5. Solve the same problem on the Cray.
6. Compute an enhancement of the problem on the Cray and document the Cray's performance relative to the campus mainframe in a written report.

Instructor feedback after reading the document from the first assignment greatly enhanced the organization and content of the final document.

6. Computational Experiments

As the lecture material was covered, students worked on their first two programming assignments. The goal was to develop a feeling for timings on the SDSU mainframe compared to accuracy of approximation schemes. Students were asked to assess how much it costs (measured in CPU time) to get a good answer (measured by true error). Since most students had little numerical analysis background, the instructor provided the original code for the "First Program" problem in Fortran and C, and the students were instructed to identify the crucial loops and insert the appropriate timing calls.

- ### First Program 1991 Course

  Run and time a Fortran code that solves a two-point boundary value problem
  y"(t) + \pi^2 y(t) = 0 ; y(0) = 0 y(1/2) = 1
  via finite differences. The given true solution is y(t) = sin (\pi t)

  The dimension of the approximation should be varied and you should identify and time the separate pieces of the solution process. You should document the performance of the SDSU mainframe on this problem and discuss the sensitivity in accuracy and timing.

  NOTE: This proved to be a somewhat confusing problem for students new to numerical analysis. There are too many sources of error. As the finite difference grid is refined, the approximation is more accurate. But the linear system that is solved becomes more poorly conditioned, thereby magnifying round-off error.

- ### First Program 1992 Course (and subsequent offerings)

  Consider the linear system A x = b, where the elements of N by N matrix A is given by a[ij] = 1/(i+j-1) (the notorious Hilbert matrix). The right-hand side, b, will be chosen so that the true solution, x, will have all components equal to 1.

  You should solve this linear system for various values of N and observe the error incurred (by computing true error, since we know the true solution should have x_i = 1) and the performance (measured by the elapsed CPU time) for the LU decomposition and subsequent solve.

- ### Second Computational Experiment (Main Project)

  Science-oriented Program to be run on SDSU Mainframe and subsequently on the Cray C90.

The main project is crucial to this course. Students will use the Cray to run this project, and the instructor did not want them to squander Cray resources before they become familiar with their particular science problem. Students are allowed to pick from a selection of problems provided by the instructor, or they can solve a scientific problem from their particular backgrounds or work environments (with approval of the instructor).

Good sources for problem statements are:

- **Computing Applications to Differential Equations: Modeling in the Physical and Social Sciences**, by J.M.A. Danby; Reston Publishing, 1985.
- **Numerical Methods and Software** by Kahaner, Moler and Nash; Prentice-Hall Publishers, 1989.

Topics to be covered in your report:

3. Your write-up should have a self-contained statement of the problem. The reader should not have to read your code to find out what equations you are working with, or what the specific problem is that you are solving.
4. Give a complete reference to where the problem came from.
5. Define your measure of work so that comparisons can be made when you run on the Cray. You can't talk about "faster" or "better" without a specific measure of performance.
6. Discuss conclusions drawn from the science of the problem itself. What is the "science" story revealed by the original problem? Why was this problem solved?
7. Discuss conclusions drawn from the performance of your code for solving the problem.
8. You should carefully organize the results. A summary of pertinent results for both the "science" of the problem and the "performance" of the program should be presented. Optionally, include an appendix for more detailed results.

Lessons Learned From Teaching CS 575

The writing assignments for CS 575 became more focussed after teaching the course the first time. Students needed to be taught how to put together a technical document as well as given hints on how to manage the large amounts of data that their computational experiments tended to produce. Giving students the specific list of items to cover in their reports (above) helped focus the writing.

Care had to be taken with the background that could be assumed for the students in the course. Although it would be beneficial to the instructor to have a Numerical Analysis prerequisite, it was felt that this would significantly reduce the number of enrolled students. Instead the attitude of the instructor is to cover in lecture the specific topics needed such as behavior of round-off error, sensitivity for solving linear systems of equation as revealed by the estimate of the condition number and the vector-oriented solution of Initial Value Problems using a simple Runge-Kutta-Fehlberg technique (with motivation from examining Euler's Method in a vector formulation). See the **NOTE** above from the first programming assignment in the 1991 course. Too many sources of errors along with being new to the behavior of floating point arithmetic was too much for the students.

Although the language of choice for the majority of current HPC users is still Fortran, undergraduate students have a strong preference for C. The Cray Training Report (**TR-OPT 1.0(E)) CF77 and Cray Standard C Optimization for Parallel-Vector Systems**) proved invaluable for examples in lecture since it presents all examples in both Fortran and C and students could then choose to focus on the example language they were most comfortable with.

The introductory programming projects were made available in Fortran as well as in C coded drivers that interface and call the Fortran subroutines. These codes are available on line along with the lecture notes from the course, as mentioned previously.

The programming requirement for CS 575 tended to encourage computer science students to enroll in the course, but discouraged students in other science disciplines to enroll. After speaking with faculty in the other departments, it was apparent that a course introducing programming with an orientation towards scientific problem solving was needed. This is a different point of view than the beginning programming language course for computer science majors which at SDSU uses Pascal and leads to the data structures course.

Evolving Curricula at SDSU

A new course was designed, **CS 205 Introduction to Computational Programming**. This course appeared in the 1994/95 SDSU catalog and was taught by Stewart in Fall 1994. This is a sophomore level course whose prerequisite is one year of calculus. The computing environment used is MATLAB from MathWorks, Inc. The C programming language is introduced via the MATLAB script writing capability. The process of conceptualizing problems in vector notation is natural for MATLAB, providing a smooth transition to the vector-oriented computing highlighted in the CS 575 Supercomputing course. Computer graphics are used in all class examples and computational experiments since this is an integral part of MATLAB. The recently released **Student Edition MATLAB Version 4** provides students with an individual copy of the software that runs on their home Macintosh or IBM/PC that is equivalent to the network version available on the instruction computers on campus. This course will be offerred Fall 1995 at SDSU and the materials will be made available during this time.

The rapidly evolving world of undergraduate curriculum development in high performance computing benefits from multidisciplinary work. At SDSU, there has been a great deal of cooperation with the Departments of Mathematical Sciences, Physics (which now has an undergraduate degree program in Computational Physics), Chemistry, Geology, Astronomy and more. An interesting source that documents academic programs in Computational Science is provided by Dr. C. D. Swanson at Cray Research Inc. http://www.cray.com/PUBLIC/ind/univ/Comp_Sci_Paper.html (WWW link)

Nation-wide Cooperative Programs

The author has been fortunate to be involved with educational activities that can be directly related to the curriculum development of CS 575. The URLs are provided in this paper for those who have access to the Wide World Web. These are ancillary sources that readers are encouraged to investigate, but are not crucial for this presentation therefore no local copy will be provided here.
Department of Energy's Undergraduate Computational Science and Engineering (**UCES**) http://uces.ameslab.gov/UCES/ (WWW link) This is a voluntary group of faculty who meet on roughly a quarterly basis to discuss and disseminate curricula materials. The program is coordinated by Dr. Tom Marchioro of Ames Laboratory. UCES presented their first public viewing of the archive of curricula materials at Supercomputing '94 in Washington, D.C. The CS 205 Introduction to Computational Programming course at SDSU discussed above has been included in the UCES archive.

UCES goes further than simply promoting discussions and sharing materials. In 1994 an award was establish to recognize and reward superior contributions to the field of computational sicence education. This award to both faculty and students was renewed for 1995. UCES also publishes *Computational Undergraduate Education*, and electronic magazine showcasing interesting developments in the field.

High Performance Scientific Computing program from the University of Colorado, Boulder
http://www.cs.colorado.edu/courses/materials.html (WWW link) In 1991, the author was fortunate to attend one of the summer workshops presented at the University of Colorado, Boulder by Dr. Lloyd Fosdick and Dr. Elizabeth Jessup. An innovative year long undergraduate course in High Performance Scientific Computing (**HPSC**) has been developed there and the instructional materials (both lecture notes, tutorials and laboratory exercises) are available via anonymous ftp or via their WWW site above.

Acknowledgements

I wish to acknowledge support from the San Diego Supercomputer Center for travel and summer salary that has been crucial to allowing me to attend some very interesting meetings for the past four years. My work in Computational Science Curriculum Development began with the support from Dr. Rich Hirsh: "Undergraduate Curriculum

Development in Advanced Computing", NSF/DASC Research Grant with Dan Sulzbach (PI), San Diego Supercomputer Center, 1990-93.

References

3. Cray Research Incorporated, TR-OPT (Rev. E) CF77 and Cray Standard C Optimization for Parallel-Vector Systems http://www.cray.com (Web link)
4. Danby, J.M.A., **Computing Applications to Differential Equations: Modeling in the Physical and Social Sciences**, Reston Publishing, 1985.
5. Dowd, K., **High Performance Computing** , O'Reilly Publishers, 1994, http://gnn.com/gnn/bus/ora/item/hpc.html (Web link)
6. Fosdick, L. and Jessup, E., University of Colorado, Boulder, HPC Course http://www.cs.colorado.edu/ftp/pub/HPSC/README.html (Web link) [contact Dr. Lloyd Fosdick, fosdick@cs.colorado.edu or Dr. Elizabeth Jessup, jessup@cs.colorado.edu]
7. Kahaner, D., Moler, C., and Nash, S., **Numerical Methods and Software** , Prentice Hall Publishers, 1989.
8. Marchioro, T.L., Department of Energy Undergraudate Computational Science and Engineering Program (UCES) http://uces.ameslab.gov/uces/ (Web link) [contact Dr. Tom Marchioro, tlm@ameslab.gov]
9. MathWorks Inc., publishers of MATLAB http://www.mathworks.com (Web link for MATLAB)
10. SDSC Cray Users Guide http://www.sdsc.edu/Services/Consult/Cray/CUG/CUG_intro.html (Web Link)
11. Stewart, K., CS 575 Supercomputing for the Sciences, Course at SDSU http://www.stewart.cs.sdsu.edu/cs575.html (Web link) .
12. Stewart, K., CS 205 Introduction to Computational Programming, SDSU Course http://www.stewart.cs.sdsu.edu/cs205.html (Web link)
13. MathWorks, Inc., **The Student Edition of MATLAB, Version 4 User's Guide**, Prentice Hall, 1995
14. Swanson, C.D., Cray Technical Report on HPC Education Programs, Feb. 2, 1995, http://www.cray.com/PUBLIC/ind/univ/Comp_Sci_Paper.html (Web link) [contact Dr. Swanson, cds@cray.com]
