Proceedings of the Fifth Annual International Conference on Technology in Collegiate Mathematics

William Rainey Harper College
and Northern Illinois University
Rosemont, Illinois
November 12-15, 1992

ADDISON-WESLEY PUBLISHING COMPANY
Reading, Massachusetts • Menlo Park, California • New York
Don Mills, Ontario • Wokingham, England • Amsterdam • Bonn • Sydney • Singapore
Tokyo • Madrid • San Juan • Milan • Paris

INCORPORATING HIGH PERFORMANCE COMPUTERS INTO MATHEMATICS CURRICULUM[1]

Marcin Paprzycki
University of Texas of the Permian Basin
Department of Mathematics and Computer Science
Odessa, TX 79762
m_paprzycki@utpb.pb.utexas.edu

0. Introduction

The most characteristic feature of recent years in computer development is the increasing number of high performance computers (vector and/or parallel) and their improved accessibility. These advances led to the formulation of a group of problems known as Computational Grand Challenges, by the national High Performance Computing and Communications Program [5]. Solutions to these problems will require the development of new hardware and software. It seems obvious that a large part of this task will be undertaken by computer scientists. It must not be overlooked, however, that the Grand Challenges constitute truly multidisciplinary problems and some of them (integrated fluid and structural airframe simulation, climate modelling, fluid turbulence, combustion systems, to name just a few) belong to the domain of interests of numerical analysis as well as applied and computational mathematics. In order to prepare our students to cope with these challenges, it is imperative to include elements of supercomputing[2] into mathematical curricula.

Section 1 offers suggestions how to acquire access to a supercomputer. Section 2 presents some possible ways of incorporating supercomputers into course curricula. Section 3 discusses in some detail a Cray-supported course in Numerical Linear Algebra offered at the University of Texas of the Permian Basin (UTPB).

1. Access to Supercomputers

The first question that one needs to answer when thinking about introducing any form of supercomputing into the curriculum is: How can one obtain access to a supercomputer? This question is especially pertinent to teachers in smaller institutions with limited budgets which, most likely, will not be able to afford such systems for themselves.

The first major source of access to computer facilities are NSF sponsored supercomputing centers. According to NSF [11], there existed five such centers in 1991:

- Cornell Theory Center
- National Center for Atmospheric Research
- National Center for Supercomputing Applications
- Pittsburgh Supercomputing Center
- San Diego Supercomputing Center

Recently, the Northeast Parallel Architectures Center at Syracuse University has joined the ranks. These centers usually offer access to their machines to students. In addition, faculty members of all universities can apply for free computing time on supercomputers[3].

The second source of computer time comes from regional supercomputer centers which are typically located at large state universities. In Texas, for example, two such centers are affiliated with University of Texas in Austin and with Texas A&M University. Such centers may provide supercomputer access for schools in a given state and/or region.

The final possibility is to build one's own parallel computer using a network of computers and one of the variety of recently developed parallel environments.[4] A word of caution is in place. Using this method to build one's own machine may be very time consuming and is advised only when an appropriate level of technical support is available.

2. The Uses of Supercomputers in Course Curricula

Supercomputers can be used in a variety of ways in teaching. We will concentrate on three most obvious choices: computer science courses on supercomputing, interdisciplinary surveys of possible uses of supercomputers across the natural sciences, courses in computational mathematics and/or numerical analysis incorporating elements of supercomputing.

First, supercomputers can be used in computer science courses on parallel or vector computing. A representative example of a course on vector computing is offered by Kris Stewart[5] at San Diego State University for students majoring in the sciences and engineering. The course is structured in the following way:

1. Introduction to Unix (Unicos),
2. Cray's Architecture,
3. Computer Ethics,
4. Performance Evaluation and Monitoring,
5. Impact of Cray's Architecture on Algorithms,
5. Software tools on the Cray.

Stewart used a text by Hennessy and Patterson [7]. Students are expected to complete four computer research projects which are closely related to the students' majors.

Zahira Khan [9] from Bloomsburg University has recently developed a sequence of two undergraduate courses on parallel computing using free computer time grant form the Pittsburgh Supercomputing Center. The first (introductory) course has the following structure:

1. Introduction to parallel processing,
2. Discussion of parallelism in applications,
3. Presentation of basic techniques for parallel processing,
4. Introduction of computational models for parallel processing,
5. Design and analysis of parallel algorithms.

The second (advanced) course concentrates on in-depth discussion of issues related to:

1. Languages,
2. Compilers,
3. Operating systems,
4. Interconnection Networks,
5. Heterogenous distributed systems,
6. Detailed study of parallel architectures.

The selection of specific topics and the depth of investigation of each of them is to be decided by the teacher.

Second, an alternative way of exposing (primarily undergraduate) students to supercomputing could be a survey course of supercomputer applications. Such a course could be offered to students majoring in Biology, Chemistry, Physics, Engineering and would concentrate on possible uses of supercomputers across the disciplines. The students would not be required to be highly familiar with computers although some rudimentary knowledge of mathematics (a minimum of the Calculus sequence) would be expected since mathematical tools would be used. The proposed course may have the following form:

1. Introduction to Unix,
2. Introduction to Visualization,
3. Supercomputing in Biology,
4. Supercomputing in Chemistry,
5. Supercomputing in Physics,
6. Supercomputing in Engineering,
7. Presentation of term projects.

Each of the sections 3-6 focuses on surveying tools available in a given discipline.[6] Homework assignments ought to be directed towards the solution of particular problems from a given discipline by means of introduced tools. An important part of the course would be students' own research project in which they would be expected to solve a more advanced problem from their discipline.

Third, supercomputers can be used to support a course in numerical analysis/computational mathematics in a variety of ways: from offering a whole new degree in Computational Mathematics (such a program is currently being developed in the Center for Numerical Analysis of the University of Texas in Austin[7]), through adding a new concentration to an existing degree in computer science (School of Computer and Information Science, Syracuse University), to using supercomputers to support and modify existing courses. As an example of the latter approach, we shall discuss in some detail a course in Numerical Linear Algebra offered at UTPB.

3. Numerical Linear Algebra at UTPB

The course was originally offered as a senior level elective for both mathematics and computer science majors and was to be a typical Numerical Analysis survey course. Since UTPB is a part of the University of Texas System it has direct access to the Cray Y-MP 8/864 in the Center for High Performance Computing in Austin. This fact has influenced the decision to change the focus of the course to numerical linear algebra and in particular to the solution of linear systems. Since for some students this could have been the only exposure to numerical analysis the reference textbook by Kahaner, Moller and Nash [8] was used. It was hoped that students will keep the book for possible future reference. The class lectures were partly based on Stewart [13] and Allen, Pruess and Shampine [1]; some exercises were drawn from Grandine [6]. The course outline can be summarized as follows:

1. Introduction to Unix (Unicos) and practical tips on running jobs on the Cray
2. Basic hardware issues of a Cray type architecture
3. Computer Arithmetic
4. Review of Linear Algebra
5. Review of level 1, 2 and 3 BLAS [2, 3, 10]
6. Solution of linear systems using Gaussian Elimination
7. Some theoretical issues in stability and conditioning of such a solution.

Part 1 was necessary because most of our students are not familiar with Unix. It was also indispensable to introduce students to specific details of submitting jobs to the Cray from the front-end Convex 220. As an introductory exercise, students ran six versions of matrix multiplication and observed their performance characteristics.[8] A variety of matrix sizes was used in such a way that students were forced to run into memory bank conflicts. This led naturally to in-depth studies of Cray's hardware architecture and its possible effects on the performance of the program.

A standard discussion of computer arithmetics was supported by homework assignments based on Grandine's projects 1, 2 and 3 [6, pp. 9-13]. Each project was realized on both Cray and our local mainframe (VAX 8200). This allowed for some interesting comparisons between computer arithmetics on different hardware platforms.

The next part of the course was a review of linear algebra (based primarily on Stewart [13, Chapters 1 and 3]), which provided students with the tools used when the stability results of Gaussian Elimination are introduced. This part of the course requires a very careful selection of the material. During the actual course we spent too much time reviewing linear algebra and, as a result, there was not enough time left for the remaining subjects that we have originally planned to cover.

The importance of the results obtained by the researchers from the LAPACK project[9] and the growing acceptance of the BLAS (Basic Linear Algebra Subroutines) standard as the way of writing codes in linear algebra led us to introducing the students to the basic ideas behind level 1, 2 and 3 BLAS. As an exercise students used calls to level 1, 2, and 3 BLAS to perform matrix multiplication (six versions with level 1 BLAS, three versions with level 2

BLAS and one version with level 3 BLAS). Since the Cray's BLAS kernels are coded in the Cray Assembly Language their usage led to significant performance increases. This, in turn, enabled the discussion of the possible impact of BLAS on the development of algorithms in linear algebra.

Next, we have concentrated our attention on solving a system of linear equations using Gaussian Elimination. Students were asked to write a code that would perform Gaussian elimination with partial pivoting and run it on the Cray checking its performance characteristics for large matrices. Due to the lack of time we were not able to discuss different (SAXPY, GAXPY and DOT [4]) variants of Gaussian elimination. Such a discussion would lead naturally to the introduction of blocked algorithms. As an exercise we would then ask students to code the basic versions of blocked algorithms using BLAS kernels and to study their performance characteristics.

Finally, some basic issues in the expected stability of the solution and its dependence on the conditioning of the matrices were discussed.

4. Final Remarks

There exist a variety of ways of employing supercomputers in the mathematics/computer science curriculum. We have discussed some of the possibilities and presented a course that was offered at the University of Texas of the Permian Basin. We believe that the course was fairly successful. Students liked its focus and really enjoyed working on the Cray. We believe that it is not so important what particular avenue to familiarizing students with supercomputing will be taken; the important point is to acquaint them with the new challenges supercomputers pose to the computational practice as they are the ones who will have to solve them.

NOTES

1. The present paper is partially based on the materials presented during a Faculty Workshop on Curriculum Development ("Supercomputing and Undergraduate Education") at the San Diego Supercomputer Center in July 1992.

2. The term "supercomputer" will be used very broadly to cover all possible kinds of vector, parallel and vector-parallel architectures.

3. According to Dan Sulzbach (Executive Director, San Diego Supercomputer Center, private communication) grant applications for limited amounts of time to support teaching are likely to be approved.

4. Since a very large number of new parallel environments are currently under development in numerous places one of the possible sources of up to date information may be the usenet discussion group comp.parallel, which is also distributed by e-mail (fpst@hubcap.clemson.edu).

5. Course description is based on the materials presented by Kris Stewart during the workshop "Supercomputing and Undergraduate Education," San Diego, July 1992.

6. The nature of the proposed course favors certain hardware platforms over others. For the course to fulfill its purpose a number of diversified computer applications must be available on a given computer. A thorough research of the existing tools on the machine one wants to use for the course is required.

7. David Kincaid discussed this program during the Permian Basin Supercomputing Conference, Odessa, March 1992.

8. For more information about using matrix multiplication in teaching computer science see [12].

9. LAPACK Project Technical Reports are available from drake@cs.utk.edu.

REFERENCES

1. Allen, R., Pruess, S., Shampine, L.F., Fundamentals of Numerical Computing, Lecture Notes -- Revised Edition, Southern Methodist University, 1987

2. Dongarra, J.J., Du Croz, J., Duff, I., Hammarling, S., A Set of Level 3 Basic Linear Algebra Subprograms, Technical Report ANL-MCS-TM57, Argonne National Laboratory, 1988

3. Dongarra, J.J., Du Croz, J., Hammarling, S., and Hanson, R.J., An Extended Set of FORTRAN Basic Linear Algebra Subprograms, *ACM Transactions on Mathematical Software*, **14** (1), 1988, 1-17

4. Dongarra, J.J., Gustavson, F.G., and Karp, A., Implementing Linear Algebra Algorithms for Dense Matrices on a Vector Pipeline Machine, *SIAM Review*, **26**, 1984, 91-112.

5. Grand Challenges 1993: High Performance Computing and Communications, A Report by the Committee on Physical, Mathematical, and Engineering Sciences, Federal Coordinating Council for Science, Engineering and Technology, Washington, 1992

6. Grandine, T.A., *The Numerical Methods Programming Projects Book*, Oxford University Press, Oxford, 1990

7. Hennessy, J.L., Patterson, D.A., *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Mateo, 1990

8. Kahaner, D., Moler, C., Nash, S., *Numerical Methods and Software*, Prentice Hall, Englewood Cliffs, 1989

9. Khan, Z.S., Incorporating Supercomputing in the Undergraduate Computer Science Curriculum, Proceedings of the Eighth Annual Eastern Small College Computing Conference, *The Journal of Computing in Small Colleges*, **8** (2), 1992, 112-121

10. Lawson, C.L., Hanson, R.J., Kincaid, D.R., and Krogh, F.T., Basic Linear Algebra Subprograms for FORTRAN Usage, *ACM Transactions on Mathematical Software*, **5** (3), 1979, 306-323

11. The National Science Foundation Supercomputing Centers 1991, edited by A. Redelfs, NSF, 1991.

12. Paprzycki, M., Matrix Multiplication: a Very Simple Program with Some Interesting Consequences, Proceedings of the Eighth Annual Eastern Small College Computing Conference, *The Journal of Computing in Small Colleges*, **8** (2), 1992, 154-162

13. Stewart, G.W., *Introduction to Matrix Computation*, Academic Press, New York, 1973