

CREATING AN UNDERGRADUATE COURSE IN PARALLEL COMPUTING USING PVM

*Chuck Pheatt
Hyung-Sun Choi*

*Emporia State University
Division of Mathematics and Computer Science
Emporia, Kansas 66801*

ABSTRACT

In recent years, courses on parallel computation have been developed and offered in many institutions as recognition of the growing significance of this topic in computer science. Introducing a course in parallel computing at a small university presents some unique problems with respect to curriculum and resources. This first course in parallel computing examines parallel methods and their applications. This paper will explore the ideas, syllabi, course materials, software, and experiences in planning for such a course. A discussion of providing a parallel computing resource using the PVM message passing software and an initial student project using the environment is also provided.

INTRODUCTION

Planning to teach an undergraduate course on parallel computing for the first time is a daunting task. Although parallel computing courses are currently offered at a number of colleges and universities, they are still relatively uncommon. Parallel computing is not a standard topic in most undergraduate computer science programs.

It was decided early in the development process that this parallel-computing offering should focus on applications of parallel methods rather than the theoretical underpinnings of the topic. This decision was motivated by the fact that the department's current computer science program has a large number of senior course offerings tending toward the theoretical. By providing a challenging sophomore or junior level course, students could build expertise in the area prior to their senior year, which is typically filled with general education course "clean-up" as well as the department's challenging software-engineering course. In addition, a course with a CS1 and possibly a CS2 prerequisite would allow interested non-majors (natural science majors with an interest in computation) to participate in the course.

As with any new course development effort, the questions of resources, syllabus, and text must be considered. Most courses in the area are designed for a graduate audience, or they may deal only with a subtopic such as parallel algorithms or parallel architectures. Time-tested resources are not an option for seeking guidance in the course development stage.

The following paragraphs discuss considerations for developing such a course. Links to many of the resources noted below are available at a companion web-site developed for this paper [1].

COURSE RATIONALE

Problems in many disciplines are rapidly becoming too large to be solved using conventional computing techniques. These problems require computational power many orders of magnitude greater than computers available today. Parallel computing represents one possible way to achieve the performance necessary to address these problems.

Writing parallel programs is in general a unique challenge to programmers. Classic computer science training emphasizes skills that are sequential in nature. Undergraduate computer science students must be introduced to the parallel methods of computing to produce the results that will be essential.

Parallel programming differs from traditional programming in that an understanding of the computer architecture is many times necessary to develop practical parallel algorithms.

This course will introduce students to parallel computing in general. Students will be introduced to the ideas and concepts of parallel computing, parallel computing hardware, and programs. They will be required to write a number of programs to develop experience in parallel programming.

COURSE OBJECTIVES

The course has the following major objectives:

- To introduce the student to the fundamental issues in design and development of parallel programs.
- To give the student a good working knowledge of the techniques used in developing effective parallel programs.
- To give the student hands-on experience writing, running, and measuring the performance of parallel programs.
- To provide a survey of parallel hardware and software computing.

Topics will include an introduction to parallel architectures, parallel algorithms, parallel languages, parallel algorithm design, message passing, analysis, and debugging.

TEXT SELECTION

An excellent summary (albeit dated) of applicable texts is shown in [2]. There are several recent texts that are directly applicable to the course with the following three presenting material at an appropriate level for the course objective:

- Wilkinson and Allen, *Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers* [3]. Requires no prerequisites in parallel programming, as it assumes only C programming knowledge. Concentrates on parallel programs that can be executed on networked workstations. Has a comprehensive support web site, including examples, assignments, and instructional materials for using the MPI and PVM.
- Kumar, et al, *Introduction to Parallel Computing: Design and Analysis of Parallel Algorithms* [4]. Intended for senior undergraduate or beginning graduate students, examines topics such as sorting and graph algorithms, discrete optimization techniques, and scientific computing applications.
- Crichlow, *Introduction to Distributed and Parallel Computing* [5], assumes a basic knowledge of programming, data structures, file design, operating systems, and computer organization.

Of these selections, the Wilkinson and Allen text provides a comprehensive hands-on approach to the topic most in-line with the course objectives.

DEFINING PARALLEL COMPUTING RESOURCES

Several alternatives were considered for a parallel resource for students to use. Several simulation systems were considered:

- Proteus [6] is an execution-driven parallel-computer simulator which simulates message-passing.
- SimOS [7] is a machine simulation environment designed for the efficient and accurate study of both uniprocessor and multiprocessor computer systems. SimOS simulates computer hardware in enough detail to run an entire operating system.
- C-Linda [8] is a co-ordination language. It uses the MIMD (multiple instructions, multiple data) model of parallel computing. Though C-Linda is not based on message passing or shared memory, it has characteristics of both.
- Parallax [9, 10] is a language model allows structured and machine independent parallel programming in a high level language. Parallax programs contain a functional description of the parallel machine architecture, which they use with the parallel algorithm that refers to the architecture specification. Parallax makes certain assumptions about its abstract parallel machine that may be physically present or simulated.

In addition, two commonly available message passing systems were considered. Message passing is the method of communication based on the sending and act of receiving of messages through a network of computers following rules of protocol of communication between some processors that possess proper memory. The processes have access to local memory. Information is sent to the local memory of a process from the local memory of a remote process. In this model, the programmer is responsible for the synchronization of the tasks. PVM - Parallel Virtual Machine [11] and MPI - Message Passing Interface [12], are examples message passing libraries.

PVM is older than the MPI, having appeared in 1989 in the laboratories of the Emory University and Oak Ridge National Laboratory [13], where it was created with the objective to create and to execute applications using existing hardware (heterogeneous collection of computers). Currently version 3.4.2 is available and offers interoperability between UNIX and NT platforms.

PVM supports diverse architectures and network architectures. User programs may be written in C or Fortran and access PVM through the use of calls to PVM library routines for functions such as process initiation, message transmission and reception, and synchronization.

The first version of MPI was published in 1994 and updated in June of 1995. It is the product of the Message Passing Interface Forum (MPIF) [14], with participation from over 40 organizations. The forum met from November 1992 to April 1994 to discuss and define a set of library interface standards for message passing.

The goal of MPI is to develop a widely used standard for writing message-passing programs. The interface attempts to establish a practical, portable, efficient, and flexible standard for message passing. MPICH [15] is a freely available, portable implementation of MPI. The current version of MPICH is 1.2.0 and was released on December 2, 1999.

Investigation of the resource alternatives found that the PVM and MPI approaches are the most commonly used tools in graduate course offerings and have a wealth of example applications available. Comparisons of the two approaches are well summarized in [16] and [17]. The major consideration in which of the approaches to select included:

- Ability to run on both Unix and NT platforms.
- Availability of low cost [11] or on-line documentation [13].

- Compatibility with available compilers (C/C++).

Based on these considerations, PVM was selected for implementation and use in the course.

PVM IMPLEMENTATION

It was decided to install PVM on two computer clusters. One cluster would be used for programming development and debugging and the second for limited "production" runs.

- Development cluster - due to recent upgrades to faculty and support computers at the university, a number of 486-66MHz computers were available. Red Hat Linux and PVM were installed on eighteen machines. These machines were also connected to a dedicated 16Mbps token ring network attached to the Internet with a permanent 56Kb connection. A minimum of Unix utilities were installed to maximize available space for PVM application programs.
- Production cluster - twenty-two student laboratory machines (400-450MHz) running Windows NT were configured to run PVM. These machines were networked with 100Mbps switched Ethernet connections. Additional configuration included installation of the PVM libraries and daemon as well as NT compatible Unix utilities [18] to allow for inter-machine communication. This cluster is used during laboratory "off peak" hours since PVM use significantly slows concurrently run user applications.

Installation and utilization on the development cluster was uneventful. The production cluster installation was more problematic. Significant effort was expended in establishing the appropriate compiler and library parameters for Visual C++ 6.0 development environment. Details on the entire PVM configuration and setup are summarized in [1].

COURSE PROTOTYPE AND PROJECT

Prior to offering the course to students at large it was decided to offer the course as an independent study to aid in the debugging process. Included in this process was the development of a project that could be used in the course as a capstone experience. It was desired that this project would address the following issues:

- Fully address the considerations that must be taken into account when constructing a parallel implementation of a problem's solution in the context of the PVM environment.
- Provide an opportunity for students to sharpen their C/C++ programming skills. It should also be able to be implemented with a moderate amount of coding.
- Utilize a type of problem that is familiar to students with a CS1 and possibly a CS2 prerequisite background.
- Be a problem that does not have a limited range of possible solutions implemented in either a sequential or parallel fashion. This factor is important if it is of interest to promote competition among students in finding a problem solution.
- The problem should be scalable. That is, a range of problems with varying difficulty should be available.

Based on these requirements, it was decided to use the traveling salesperson problem (TSP) with solutions generated using a genetic algorithm (GA) [19] approach implemented in a parallel fashion. Although this particular approach to this problem has been implemented in other venues [20, 21], it is sufficiently unusual to provide opportunities to break new ground.

The GA approach creates a population of problem solutions and applies the operations of reproduction, crossover and mutation to create successive generations of solutions.

Solutions are probabilistically selected for inclusion in successive generations based on their fitness in solving the problem. For the TSP problem, fitness is simply the distance among the cities in the problem set.

Using this approach, students can first implement TSP using the GA approach. Once a sequential implementation is successfully readied, it may then be migrated to the parallel environment using a SPMD (single program, multiple data) configuration using the host-node (also known as master-slave) model or the node-only PVM models. Students may further enhance their programs by allowing instances of the program to exchange problem solutions simulating population migration among pockets of individuals.

Prototyping the course in this fashion allowed many of the bugs that typically appear in new courses to be resolved before bringing the course on-line.

CONCLUSIONS AND FUTURE WORK

Development of a parallel computing course for undergraduates has provided an opportunity to become familiar with the current state of the field and provide students with an offering that should encourage further study in the field. Challenges in developing the computing resource for the course as well as the course syllabus were in themselves a significant learning experience for the authors.

As the course continues to develop, the authors will investigate implementing the MPI approach in the Windows NT environment. The current trend in the literature appears to favor the MPI approach.

REFERENCES

1. Pheatt, C. Choi, H., Creating An Undergraduate Course In Parallel Computing Using Pvm web site, <http://mathcssun.emporia.edu/pvm/>.
2. Worland, P. B., "Resources for an Undergraduate Course on Parallel Computing", Tenth Annual Midwest Computer Conference, Loyola University Chicago, Friday, 1996.
3. Wilkinson, B. and Allen, M., *Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers*, Prentice Hall, Upper Saddle River, NJ, 1998.
4. Kumar, V., Grama, A., Gupta, A. and Kumar, G. K., *Introduction to Parallel Computing: Design and Analysis of Parallel Algorithms*, Addison-Wesley Publishing Co., Reading, MA, 1994.
5. Crichlow, J., *Introduction to Distributed and Parallel Computing*, Prentice Hall, Upper Saddle River, NJ, 1988.
6. Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing. Scientific and engineering computation.* MIT Press, Cambridge, MA, 1994.
7. Message Passing Interface Forum. "MPI: A message-passing interface standard". *International Journal of Supercomputer Applications*, 8(3/4): 159-416, 1994.
8. Oak Ridge National Laboratory, Computer Science & Mathematics Division, Parallel Virtual Machine web site, http://www.epm.ornl.gov/pvm/pvm_home.html.
9. Argonne National Laboratory, Mathematics and Computer Science Division, MPICH-A Portable Implementation of MPI web site, <http://www-unix.mcs.anl.gov/mmpi/mpich/>.
10. Argonne National Laboratory, Mathematics and Computer Science Division, The Message Passing Interface (MPI) Standard web site, <http://www-unix.mcs.anl.gov/mmpi/>.
11. Louisiana State University, Department of Electrical & Computer Engineering, Proteus web site, <http://www.ee.lsu.edu/koppel/proteus.html>.
12. Stanford University, Computer Science Department, The Complete Machine Simulator (SIMOS) web site, <http://simos.stanford.edu>.

13. Scientific Computing Associates, Linda web site, <http://www.sca.com/linda.html>.
14. Queensland Parallel Supercomputing Foundation, Parallaxis web site, <http://www.qpsf.edu.au/software/parallaxis.html>.
15. Universität Stuttgart, ftp site, <ftp://ftp.informatik.uni-stuttgart.de/pub/parallaxis/>.
16. Gropp, W. and Lusk, E., "Why are PVM and MPI so different?" Mathematics and Computer Science Division, Argonne National Laboratory, preprint ANL/MCS-P667-0697, 1997.
17. Gropp, W. and Lusk, E., "PVM and MPI Are Completely Different," Mathematics and Computer Science Division, Argonne National Laboratory, preprint ANL/MCS-P737-1198, 1998.
18. Unix Remote Shell Daemon (RSHD), which services the RCP and RSH commands. Available from Denicomp Systems, <http://www.denicomp.com/>.
19. Holland, J.H., *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, MI, 1975.
20. Ward SystemsGroup, Inc, GeneHunter software web site, <http://wardsystems.com/download.htm>.
21. Johns, J., Using a GA to solve the Traveling Salesman Problem web site, <http://www.cs.rpi.edu/~skolnick/ga/students/johnsj/exi.html>