Software Reuse in High Performance Computing

Shirley Browne
University of Tennessee
107 Ayres Hall
Knoxville, TN 37996-1301
Tel: (615) 974-5886
Fax: (615) 974-8296
Email: browne@cs.utk.edu

Jack Dongarra
University of Tennessee and Oak Ridge National Laboratory
Email: dongarra@cs.utk.edu

Geoffrey Fox
Syracuse University
Email: gcf@npac.syr.edu

Ken Hawick
Syracuse University
Email: hawick@npac.syr.edu

Tom Rowan
University of Tennessee and Oak Ridge National Laboratory
Email: rowan@msr.epm.ornl.gov

Abstract

Although high performance computing architectures in the form of distributed memory multiprocessors have become available, these machines have not achieved widespread use outside of academic research environments. The slowness in adopting high performance architectures appears to be caused by the difficulty and cost of programming applications to run on these machines. Economical use of high performance computing and subsequent adoption by industry will only occur if widespread reuse of application code can be achieved. To accomplish this goal, we propose strategies for achieving reuse of application code across different machine architectures and for using portable reusable components as building blocks for applications.

Keywords: high performance computing, parallel and distributed computation, linear algebra, distributed memory multiprocessor

Workshop Goals: Advance state of reuse of parallel software

Working Groups: Tools and environments, Education

1 Background

We have been involved in the development of the Netlib mathematical software repository (`http://www.netlib.org`) and of much of the reusable software available from Netlib and the National HPCC Software Exchange (NHSE) (`http://www.netlib.org/nse/`). The NHSE is a recent effort to construct a distributed virtual repository of all the software produced by the High Performance Computing and Communications (HPCC) program. As part of the NHSE, we have developed the InfoMall technology transfer program (`http://www.infomall.org/`) which identifies reusable HPCC technologies and makes them available to industry. In the area of parallel computing, we have carried out pioneering work in programming applications on parallel computers and in determining how to make the resulting codes portable across different parallel architectures.

2 Position

High performance computing involves the use of parallel computing systems to solve computationally intensive applications. Some of these applications, termed Grand Challenges, address fundamental problems in science and engineering that have broad economic and scientific impact. Examples include global climate modeling, materials simulation at the molecular level, coupled structural and airflow simulation for aircraft, and mapping of the human genome. The computational requirements of these problems have pushed the physical limits of how fast processors can operate and have focused effort on increasing the number of processors in a multiprocessor. Consequently, emphasis is being placed on developing parallel algorithms and software that are *scalable*, meaning that they continue to perform their tasks efficiently as the number of processors is increased.

Parallel computing will realize its full potential only if it is accepted and adopted in the real world of industrial applications. Cost-effective parallel computing will require that the specialized hand coding of parallel programs gives way to widespread reuse of parallel software. Although specialized hand coding can achieve peak performance on the target architecture, a small loss in performance will be acceptable to achieve a more productive software development environment.

Reuse of parallel software involves the following aspects:

- Portability and scalability of application codes.

  Because writing parallel programs requires substantial effort and investment, it is unacceptable to rewrite for every current and future parallel architecture. Code that is portable and scalable, however, will run with high efficiency on almost all current and anticipated future parallel architectures.

- Use of fundamental building blocks.

  A large number of scientific and engineering applications rely heavily on a small core of linear algebra routines. Scalable, portable, efficient, and easy-to-use versions of these routines will provide the fundamental building blocks for applications.

We propose a strategy for promoting software reuse in high performance computing that includes the following components:

1. A classification scheme for applications that maps an application to a problem class, rather than to a machine architecture, and the development of high level software systems based on this classification.

   Fox, together with other researchers, has developed a classification scheme for applications that includes the categories *synchronous*, *loosely synchronous*, *asynchronous*, and *embarrassingly parallel* [FWM94]. After a developer maps an application to a particular class, he should express that application in a programming language that has features optimized for that class, rather than for a particular machine architecture. Using a language such as Fortran 77 or C enhanced with message-passing obscures the parallelism that is inherent in a problem. Expressing a problem in a parallel language that is nearer the problem than to the target machine automatically enhances reuse because the resulting program may be more easily ported to a variety of parallel machines.

2. Development of high-level parallel languages for writing scalable, portable code.

   High Performance Fortran (HPF) [HPF94] has been agreed on as a new industry standard data parallel language. The synchronous and embarrassingly parallel application classes may be expressed straightforwardly in HPF. Researchers are working on extensions to HPF to handle the irregular structure of loosely synchronous problems[DHCF]. HPF compilers will be able to obtain good performance for the same code on a variety of parallel architectures [BCF$^+$]. Asynchronous problems cannot be expressed in a data parallel language and will require different algorithmic and software support built on top of message passing.

3. Adoption of the Message Passing Interface (MPI) portable message-passing standard.

   The MPI standard defines the syntax and semantics for a core of library routines useful for writing portable message-passing programs in Fortran 77 or C [DOSW95]. MPI forms a possible target for compilers of high-level languages such as HPF.

4. Development of a scalable linear algebra library for distributed memory multiprocessors.

   Researchers led by Jack Dongarra at the University of Tennessee are developing the ScaLAPACK library of subroutines for performing linear algebra computations on distributed memory multiprocessors [DW93, CDD$^+$95]. When completed the library will contain subroutines for performing dense, banded, and sparse matrix computations. Among the important design goals are scalability, portability, flexibility, and easy of use. All communication in a ScaLAPACK library routine is handled within the distributed Level 3 Basic Linear Algebra Subroutines (BLAS) and the Basic Linear Algebra Communication Subroutines (BLACS), so that the user is isolated from the details of the parallel implementation. Specially tuned versions of the BLAS and the BLACS may be implemented by the manufacturer of a parallel architecture to achieve near peak performance for matrix computations.

5. Dissemination of software and information related to high performance computing via the National HPCC Software Exchange (HPCC).

   The NHSE has a software submission and review procedure that classifies software submissions and evaluates them according to a set of criteria that includes scope, completeness, documentation, construction, correctness, soundness, usability, and efficiency [BDKR95]. Roadmaps to HPCC software and technologies are being developed that will assist the user in locating and understanding relevant reusable components. For example, a roadmap that illustrates the use and application of the High Performance Fortran (HPF) language has been developed (`http://www.npac.syr.edu/hpfa/`). Educational and technology transfer materials and activities are also being undertaken by the NHSE.

3 Comparison

The Archetype Working Group at CalTech (`http://www.etext.caltech.edu/`) is developing a library of parallel program archetypes with the aim of reducing the effort required to produce correct and efficient parallel programs. A parallel program archetype is a program design strategy appropriate for a restricted class of problems. The design strategy includes the software architecture for the problem class, archetype-specific information about how to derive a program from its specification, methods of reasoning about correctness and performance, suggestions for test suites, and suggestions for performance tuning on different architectures. An archetype identifies the common computational and communication structures of a class of problems and provides development methods specific to those structures. So far archetypes have been developed for mesh computations and for spectral methods. Archetypes are proposed as an alternative to reuse libraries. The Archetype Group argues that reuse libraries are inadequate for parallel programming because they are language and run-time system specific and because of difficulties with parallel composition of library procedures. Our approach has been to develop portable subroutine libraries in Fortran, which is the predominant programming language for scientific computing, and to isolate the applications programmer from the details of the parallel implementation.

The High Performance C++ (HPC++) project is attempting to extend industry standard object oriented software ideas to the world of massively parallel computing (`http://www.extreme.indiana.edu/hpc++/index.html`). Parallel extensions to the C++ programming language include pC++, a data parallel programming language that combines HPF style programming with object oriented techniques, and CC++, a task parallel extension to C++ suited to both distributed and parallel applications. The HPC++ project is also working to extend the Object Management Group's CORBA and IDL specifications to support application servers running on massively parallel computers. Such extensions will support language interoperability between C++ based parallel programming systems and MPI implementations. Work on parallel object oriented programming languages complements our work on data parallel problem architectures and on high performance linear algebra libraries. We have developed LAPACK++, an object-oriented C++ extension of LAPACK (`http://www.netlib.org/c++/lapack++`). The object oriented approach provides an extensible framework for incorporating extensions of LAPACK, such as ScaLAPACK++ for distributed memory architectures.

Researchers at Rice University are developing the D System, which is a set of tools for machine-independent data parallel programming (`http://www.cs.rice.edu/fortran-tools/DSystem/DSystem.html`) [ACG$^+$94]. The tools support development of programs in Fortran D, an abstract, machine-independent parallel programming language. The D System editor allows the programmer to interactively parallelize existing Fortran code, and the data mapping assistant helps the programmer choose good decompositions for data structures. Because data parallel compilers perform aggressive program transformations, the relationship between the source program and object program may be difficult for the programmer to understand. Thus, the D System debugging and performance analysis tools support analysis in terms of the source program. Fortran D is a predecessor to HPF, and features of the D System are expected to carry over to environments for HPF and other data parallel languages. We speculate that the D System might be extended to support our problem classification scheme and to partially automate selection and use of linear algebra components.

References

[ACG$^+$94] V. Adve, A. Carle, E. Granston, S. Hiranandani, K. Kennedy, C. Koelbel, U. Kremer, J. Mellor-Crummey, C. W. Tseng, and S. Warren. Requirements for data-parallel programming environments. *IEEE Transactions on Parallel and Distributed Technology*, 2(3):48–58, Fall 1994. Also available at `http://softlib.rice.edu/CRPC/softlib/TRs_online.html` as CRPC-TR944378-S.

[BCF$^+$] Z. Bozkus, A. Choudhary, G. Fox, T. Haupt, and S. Ranka. Compiling HPF for distributed memory MIMD computers. Technical Report SCCS-507, Northeast Parallel Architectures Center, Syracuse University. Available at `http://www.npac.syr.edu/techreports/`.

[BDKR95] Shirley Browne, Jack Dongarra, Ken Kennedy, and Tom Rowan. Management of the NHSE – a virtual, distributed digital library. In *Digital Libraries '95*, Austin, Texas, June 1995. Also available as University of Tennessee Technical Report UT-CS-95-287 at `http://www.netlib.org/tennessee`.

[CDD$^+$95] J. Choi, J. Demmel, I. Dhillon, J. Dongarra S. Ostrouchov, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. ScaLAPACK: a portable linear algebra library for distributed memory computers. Technical Report UT-CS-95-283, University of Tennessee, March 1995. Available at `http://www.netlib.org/tennessee/`.

[DHCF] Kivanc Dincer, Kenneth Hawick, Alok Choudhary, and Geoffrey Fox. High Performance Fortran and possible extensions to support conjugate gradient algorithms. Technical Report SCCS-703, Northeast Parallel Architectures Center, Syracuse University. Available at `http://www.npac.syr.edu/techreports/`.

[DOSW95] Jack Dongarra, Steve Otto, Marc Snir, and David Walker. An introduction to the MPI standard. Technical Report UT-CS-95-274, University of Tennessee, January 1995. Available at `http://www.netlib.org/tennessee/`.

[DW93] Jack Dongarra and David Walker. Software libraries for linear algebra computation on high-performance computers. Technical Report ORNL-TM-12404, Oak Ridge National Laboratory, August 1993. Available at `http://www.netlib.org/tennessee/`.

[FWM94] Geoffrey Fox, Roy Williams, and Paul Messina. *Parallel Computing Works*. Morgan Kaufmann, 1994. Accessible on-line at `http://www.infomall.org/npac/pcw/`.

[HPF94] High Performance Fortran language specification, version 1.1. Technical Report CRPC-TR92225, High Performance Fortran Forum, November 1994. Available at `http://softlib.rice.edu/CRPC/softlib/TRs_online.html`.

4 Biographies

Shirley Browne is a Research Associate and Adjunct Professor in the Computer Science Department at the University of Tennessee, where she is a member of the Netlib Development Group. Her research interests are in replicated database issues for a scalable information infrastructure and in information system support for software reuse. She received a Ph.D. in Computer Sciences from Purdue University in 1990.

Jack Dongarra holds a joint appointment as Distinguished Professor of Computer Science in the Computer Science Department at the University of Tennessee (UT) and as Distinguished Scientist in the Mathematical Sciences Section at Oak Ridge National Laboratory (ORNL) under the UT/ORNL Science Alliance Program. He specializes in numerical algorithms in linear algebra, parallel computing, use of advanced-computer architectures, programming methodology, and tools for parallel computers. Other current research involves the development, testing and documentation of high quality mathematical software. He was involved in the design and implementation of the software packages EISPACK, LINPACK, the BLAS, LAPACK, ScaLAPACK, Netlib/XNetlib, PVM/HeNCE, MPI and the National High-Performance Software Exchange, and is currently involved in the design of algorithms and techniques for high performance computer architectures. He received a Ph.D. in Applied Mathematics from the University of New Mexico in 1980.

Geoffrey Fox is an internationally recognized expert in the use of parallel architectures and the development of concurrent algorithms. He leads a major project to develop prototype high performance Fortran (Fortran90D) compilers. He is also a leading proponent for the development of computational science as an academic discipline and a scientific method. His research on parallel computing has focused on development and use of this technology to solve large scale computational problems. Fox directs InfoMall, which is focused on accelerating the introduction of high speed communications and parallel computing into New York State industry and developing the corresponding software and systems industry. Much of this activity is centered on NYNET with ISDN and ATM connectivity throughout the state including schools where Fox is leading developments of new K-12 applications that exploit modern technology. Fox is currently a Professor of Computer Science and Physics at Syracuse University and director of NPAC, The Northeast Parallel Architectures Center. He obtained his PhD from Cambridge, England and spent 20 years at Caltech where he was Professor and Associate Provost for educational and academic computing.

Ken Hawick is a research scientist at the Northeast Parallel Architectures Center (NPAC) at Syracuse University, working in the areas of computational science and high performance computing, and in information technology for industry. He formerly headed the Numerical Simulations Group at the Edinburgh Parallel Computing Centre in Edinburgh, Scotland.

Tom Rowan is a Collaborating Scientist in University of Tennessee's Computer Science Department and Oak Ridge National Laboratory's Mathematical Sciences Section. His primary research interests are in the areas of numerical analysis and mathematical software. Much of his research has focused on developing algorithms and software for automatic detection of instability in numerical algorithms and for optimization of noisy functions. He received his Ph.D. from the University of Texas at Austin in 1990.