Numerical Libraries and Tools for Scalable Parallel Cluster Computing

Jack Dongarra, Shirley Moore and Anne Trefethen

International Journal of High Performance Computing Applications 2001 15: 175
DOI: 10.1177/109434200101500210

NUMERICAL LIBRARIES AND TOOLS FOR SCALABLE PARALLEL CLUSTER COMPUTING

Jack Dongarra
Shirley Moore
UNIVERSITY OF TENNESSEE

Anne Trefethen
NUMERICAL ALGORITHMS GROUP, UK

The International Journal of High Performance Computing Applications,
Volume 15, No. 2, Summer 2001, pp. 175-180

1 Introduction

For cluster computing to be effective within the scientific community, it is essential that there be numerical libraries and programming tools available to application developers. Cluster computing may mean a cluster of heterogeneous components or hybrid architecture with some symmetric multiprocessor (SMP) nodes. This is clear at the high end (e.g., the latest IBM SP architecture) as well as in clusters of PC-based workstations. However, these systems may present very different software environments on which to build libraries and applications and indeed require a new level of flexibility in the algorithms if they are to achieve an adequate level of performance. We will consider here the libraries and software tools that are already available and offer directions that might be taken in the light of cluster computing.

2 Background and Overview

There have been many advances and developments in the creation of parallel code and tools for distributed-memory machines and likewise for SMP-based parallelism. In most cases, the parallel, Message-Passing Interface (MPI)–based libraries and tools will operate on cluster systems but may not achieve an acceptable level of efficiency or effectiveness on clusters that comprise SMP nodes. Little software exists that offers the mixed-mode parallelism of distributed SMPs. It is worth considering the wealth of software available for distributed-memory machines, since for many cases this may be entirely suitable. Beyond that, we need to consider how to create more effective libraries and tools for the hybrid clusters.

The underlying technology on which the distributed-memory machines are programmed is that of MPI (Snir et al., 1996). MPI provides the communication layer of the library or package, which may or may not be revealed to the user. The large number of implementations of MPI ensures portability of codes across platforms and, in general, the use of MPI-based software on clusters. The emerging standard of OpenMP (Dagum and Menon, 1998) is providing a portable base for the development of libraries for shared-memory machines. Although most cluster environments do not support this paradigm globally across the cluster, it is still an essential tool for clusters that may have SMP nodes.

3 Numerical Libraries

The past few years have seen continued architectural change in high performance computing. The upside of

"The past few years have seen continued architectural change in high performance computing. The upside of this development is the continued steep growth in peak performance, but the downside is the difficulty in producing software that is easy to use, efficient, or even correct."

this development is the continued steep growth in peak performance, but the downside is the difficulty in producing software that is easy to use, efficient, or even correct. We will review a list of these challenges.

A major recent architectural innovation is clusters of shared-memory multiprocessors referred to as a *constellation*. This is to avoid confusion with the term *cluster*, which usually is used in the context of a group of PCs connected through a switched network. A cluster can include as its nodes small multiprocessor SMPs. A constellation integrates nodes of large SMPs. The distinction is where the majority of parallelism exists. If it is at the top clustering level (the message-passing domain), then the system is a cluster. If it is at the bottom SMP level (the shared-memory domain), then the system is a constellation. A constellation incorporates more processors per SMP node than the number of nodes integrated at the system level. Constellations are the architectures of the ASCI machines and promise to be the fastest general-purpose machines available for the next few years, in accordance with the industrial trend to invest most heavily in large market sectors and use the same building blocks to service the smaller high-end market.

It is the depth of the memory hierarchy with its different access primitives and costs at each level that makes constellations more challenging to design and use effectively than their SMP and massive parallel processor (MPP) predecessors. Ideally, a programmer should be able to produce high performance parallel code on a constellation using an abstract programming model that is not dependent on implementation details of the underlying machine. Because users may have different notions of what is acceptable high performance, we expect the layered programming model to allow the underlying machine to be exposed at varying levels of detail. This requires good communication libraries, data structure libraries, and numerical algorithms, which we propose to build.

"Although constellations will be the dominant architecture for high-end computing, most programs will be from either SMP or MPP versions, and even new code developments will likely be done on smaller machines."

Although constellations will be the dominant architecture for high-end computing, most programs will be from either SMP or MPP versions, and even new code developments will likely be done on smaller machines. Therefore, users need a uniform programming environment that can be used across uniprocessors, SMPs, MPPs, and constellations. Currently, each type of machine has a completely different programming model. SMPs have dynamic thread libraries with communication through shared memory. MPPs have single-program multiple-data parallelism with message-passing communication (e.g., MPI). Constella-

tions typically have the union of these two models, requiring that the user write two different parallel programs for a single application or else treat the machine as "flat," with a corresponding performance loss. Well-designed libraries can hide some of these details, easing the user's transition from desktop to SMP to constellations.

In addition to constellations, architectures that are likely to be important are distributed networks and IRAM. IRAM stands for intelligent RAM and is an example of PIM, or processor in memory. IRAM consists of a chip containing both processors and a substantial memory and is a foreseeable architectural change as devices get smaller and more integrated. Distributed networks, which can be and are used now, have very slow and even unreliable access to remote memories, whereas IRAM promises to significantly flatten the memory hierarchy, with the on-chip memory effectively acting like an enormous cache. This profusion of architectural targets makes software development for scientific computing challenging.

One common feature is to build libraries that are *parameterized* for the architectural features most influencing performance. We are used to dealing with cache sizes, latency, and bandwidth in our use of performance modeling in previous work, but the complexity of constellations and other architectures presents new challenges in this regard.

Another architecturally driven algorithmic opportunity arises on the Pentium family of processors, which are not only widely used on desktops but also comprise the ASCI Red machine and will be widely used in smaller clusters as inexpensive computing platforms. This ubiquity of Intel platforms leads us to ask how to exploit special features of the Intel architecture to do better high performance computing. In addition to excellent support for IEEE standard 754 floating arithmetic, the basic arithmetic is done to 80-bit precision rather than 64 bit. A variety of algorithms perform more quickly and/or more accurately by using these features. These algorithms can be encapsulated within standard libraries and so do not require user sophistication or even awareness for their use.

Another challenge from the proliferation of computing platforms is how to get high performance from computational kernels such as matrix-matrix multiplication, matrix-vector multiplication, fast Fourier transforms (FFTs), and so on. There are systems such as ATLAS, PhiPac, and FFTW that use a sophisticated search algorithm to automatically find very good matrix-multiply kernels for RISC workstations with C compilers that do good register allocation and basic instruction scheduling; using this approach, one can produce matrix-multiply and FFT routines that are usually faster than the hand-tuned codes from IBM, SGI, and some other manufacturers.[1]

4 Current State of the Art

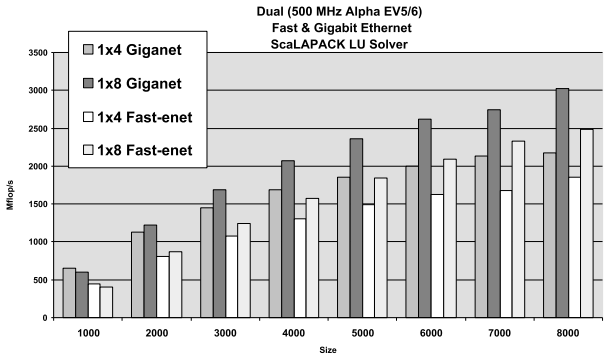
Of the hundred or more parallel numerical packages available, some provide a conventional library interface to routines written in C, Fortran, or C++. Others provide more of a parallel environment for the application developer. At the moment, few if any mix distributed- and shared-memory parallelism.

Recent surveys on parallel numerical analysis software (Allan, Hu, and Lockey, 1999; Eijkhout, 1997; Dongarra, 2000) include approximately 50 different libraries and packages for parallel architectures. All of these would be suitable for cluster computing, although they may not be fully efficient for particular configurations. The software discussed in the report forms a subset of all parallel packages that are available either commercially or distributed on Netlib[2] or on the National HPCC Software Exchange, NHSE.[3]

As one might expect, most available packages in this area are in linear algebra. Both direct and iterative solvers are well represented. Direct solver packages include ScaLAPACK (Blackford et al., 1997) and PLAPACK (Alpatov et al., 1997), both of which are based on the parallelization of LAPACK (Anderson et al., 1995) but by different approaches. The graphs in Figure 1 show good scalability of the ScaLAPACK LU solver on Alpha clusters with fast interconnection networks.

Iterative solver packages include Aztec (Hutchinson et al., 1998), Blocksolve (Jones and Plassman, 1992), and PSPARSLIB (Saad and Sosonkina, 1998). Each of these provides a selection of iterative solvers and preconditioners, with MPI providing the underlying communications layer. Similarly, there are packages and libraries for eigenvalue problems, including PARPACK (Lehoucq, Sorensen, and Yang, 1998) and PeIGS (Elwood, Fann, and Littlefield, 2000)—the first based on Arnoldi iterations and the second on Cholesky decomposition— both using MPI.

Other areas covered by existing parallel libraries include optimization and PDE solvers. All of the libraries mentioned above are available from Netlib. Commercial products are provided by many of the machine vendors, and NAG provides a commercial, supported, general parallel library (*The NAG Parallel Library Manual*, 1997) based on MPI and also an SMP library (Salvini and Waniewski, 1996; *The SMP Library User Manual*, 1997) based on OpenMP.



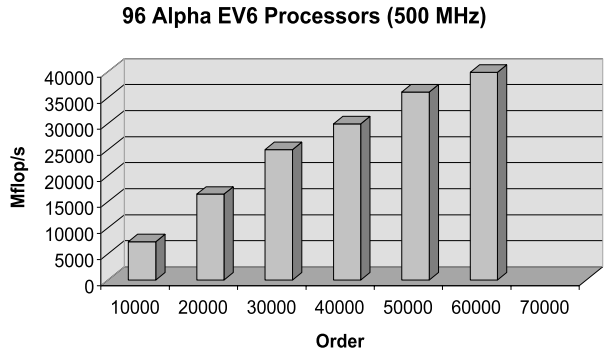


Fig. 1 Performance of ScaLAPACK LU Solver on Alpha clusters

"Scalable parallel computing on PC clusters requires the use of a message-passing system such as MPI, although OpenMP and other forms of thread-based parallelism may also be used on SMP nodes."

The libraries that have been mentioned so far all have a traditional library interface. One of the packages that offers parallel functionality in the setting of an environment is PETSc (Balay et al., 1996). PETSc provides an object-based interface to C-coded algorithms. In this case, the application developer does not need to be aware of the message-passing or underlying mechanisms. PETSc provides a set of tools for solving PDE problems, including iterative methods and other underlying functionality. This requires that the users develop their application in the PETSc style rather than calling a particular routine from PETSc. PETSc provides an object-oriented interface that is also the natural interface for C++-based packages such as ISIS++ and ELLPACK (Weerawarana et al., 1994).

It is clear that there is a wealth of software available for clusters; however, as noted above, this software is in general either developed for heterogeneous distributed-memory architectures or SMP machines. We still have some distance to travel before we can provide effective and efficient numerical software for the general cluster.

5 Future Work

To achieve transparent cluster computing requires algorithms that can adapt to the appropriate configuration to match the cluster requirements. Recent studies show that there are benefits to be gained by rethinking the parallelization issues combining distributed- and shared-memory models (Salvini, Smith, and Greenfield, 1998). It is unlikely that libraries that provide this effective mixed-mode parallelism will be available for some time. It is more likely that libraries that are inherently designed for distributed memory will be adapted to have appropriate SMP-based kernels. This is likely to provide the natural progression to general numerical libraries for clustered computing.

6 Program Development and Analysis Tools

Rapid improvements in processor performance and multiprocessing capabilities of PC-based systems have led to widespread interest in the use of PC clusters for parallel computing. The two major operating systems that support multiprocessing on such systems are Linux and Windows NT. Scalable parallel computing on PC clusters requires the use of a message-passing system such as MPI, although OpenMP and other forms of thread-based parallelism may also be used on SMP nodes. Programming languages of in-

terest for scientific computing on PC clusters include Fortran, C, and C++. Appendix B presents a survey of available program development and analysis tools for PC cluster environments. These tools include compilers and preprocessors, MPI implementations, and debugging and performance analysis tools, as well as integrated development environments (IDEs) that combine these tools. Some IDEs are developed entirely by one vendor or research group, while others are designed to work together with third-party tools (e.g., with different compilers and MPI implementations).

7 Conclusions

A good base of software is available to developers now, both publicly available packages and commercially supported packages. These may not in general provide the most effective software, but they do provide a solid base from which to work. It will be some time in the future before truly transparent, complete efficient numerical software is available for cluster computing. Likewise, effective programming development and analysis tools for cluster computing are becoming available but are still in early stages of development.

BIOGRAPHIES

Jack Dongarra earned a B.S. in mathematics from Chicago State University in 1972. A year later, he finished an M.S. in computer science from the Illinois Institute of Technology. By this time, he was already involved in the EISPACK project producing high-quality, portable, Fortran implementations of state-of-the-art algorithms for numerical linear algebra. He formally received a Ph.D. in applied mathematics from the University of New Mexico in 1980. He worked at the Argonne National Laboratory until 1989, becoming a senior scientist. He now holds an appointment as university distinguished professor of computer science in the Computer Science Department at the University of Tennessee and is an adjunct R&D participant in the Computer Science and Mathematics Division at Oak Ridge National Laboratory (ORNL) and an adjunct professor in computer science at Rice University. He specializes in numerical algorithms in linear algebra, parallel computing, use of advanced-computer architectures, programming methodology, and tools for parallel computers. His research includes the development, testing, and documentation of high-quality mathematical software. He has contributed to the design and implementation of the following open-source software packages and systems: EISPACK, LINPACK, the BLAS, LAPACK, ScaLAPACK, Netlib, PVM, MPI, NetSolve, and ATLAS. He has published approximately 200 articles, papers, reports, and technical memoranda, and he is the coauthor of several books. He is a fellow of the AAAS, the ACM, and the IEEE.

Shirley Moore received a Ph.D. in computer sciences from Purdue University in 1990. She is the academic lead for programming tools in the Programming Environment and Training (PET) component of the Department of Defense High Performance Computing Modernization Program at the Army Research Laboratory and Aeronautical Systems Center Major Shared Resource Centers. As part of PET, she has been involved in evaluation and technology transfer of debugging and performance analysis tools and in training users on how to use these tools. She has also worked closely with application scientists on debugging and performance tuning of parallel applications. Previously, she was the technical lead for the National High Performance Software Exchange (NHSE) project, which has the goal of promoting sharing of high performance software and tools among U.S. federal high performance computing agencies. Her current research interests are in a scalable cross-platform architecture for application performance analysis and in grid-capable debugging technology.

Anne Trefethen joined NAG Ltd. in 1997 as a senior technical consultant in the Numerical Libraries Division. She is now vice president for research and development of NAG and jointly leads the research and development component of the company and manages the work of the Numerical Libraries Division. Before joining NAG, she was an associate director of the Cornell Theory Center, where she had worked for 5 years as a researcher in parallel algorithms. From 1987 to 1991, she worked at Thinking Machines Corporation, where she was involved in the development of the first Connection Machine Scientifica Software Library.

NOTES

1. See: http://www.netlib.org/atlas/ and http://www.fftw.org/.
2. Netlib (http://www.netlib.org/).
3. NHSE (http://www.nhse.org/).

REFERENCES

Allan, R., Hu, Y., and Lockey, P. 1999. *Parallel Application Software on High Performance Computers: Survey of Parallel Numerical Analysis Software* [Online]. Available: http://www.dl.ac.uk/TCSC/Subjects/Parallel_Algorithms/lib_survey/

Alpatov, P., Baker, G., Edwards, C., Gunnels, J., Morrow, G., Overfelt, J., van de Geijn, R., and Wu, Y. 1997. *PLAPACK: Parallel Linear Algebra Libraries Design Overview, SC97* [Online]. Available: http://www.cs.utexas.edu/users/rvdg/plapack.

Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenny, A., Ostrouchov, S., and Sorenson, D. 1995. *Lapack Users' Guide: Release 2.0*. Philadelphia: SIAM.

Baker, M., and Apou, A. 2001. Middleware. *International Journal of High Performance Computing Applications* 15 (2): 136-142.

Balay, S., Gropp, W., McInnes, L.C., and Smith, B. 1996. PETSc 2.0 users' manual. Technical Report ANL-95/11 revision 2.0.17, Argonne National Laboratory.

Blackford, L., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Walker, D., and Whaley, R. 1997. *ScaLAPACK Users' Guide*. Philadelphia: SIAM.

Dagum, L., and Menon, R. 1998. OpenMP: An industry-standard API for shared-memory programming. *IEEE Computational Science & Engineering* 5 (1): 46-55.

Dongarra, J. 2000. *Freely Available Software for Linear Algebra on the Web* [Online]. Available: http://www.netlib.org/utk/people/JackDongarra/la-sw.html.

Eijkhout, V. 1997. *A Survey of Iterative Linear System Solver Packages* [Online]. Available: http://www.netlib.org/ utk/papers/iterative-survey/.

Elwood, D., Fann, G., and Littlefield, R. 2000. *Parallel Eigen Solver System User Manual* [Online]. Available:http://www.emsl.pnl.gov:2080/docs/nwchem/doc/peigs/docs/peigs3.html.

Hutchinson, S., Prevost, L., Tuminaro, R., and Shadid, J. 1998. *AZTEC Users' Guide: Version 2.0* [Online]. Available: http://www.cs.sandia.gov/CRF/aztec1.html.

Jones, M., and Plassman, P. 1992. Blocksolve V1.1: Scalable library software for parallel solution of sparse linear systems. ANL Report 92/46, Argonne National Laboratory [Online]. Available: http://www-unix.mcs.anl.gov/sumaa3d/BlockSolve.

Lehoucq, R., Sorensen, D., and Yang, C. 1998. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly-Restarted Arnoldi Methods*. Philadelphia: SIAM.

The NAG Parallel Library Manual, Release 2. 1997. Oxford, UK: Numerical Algorithms Group Ltd.

Saad, Y., and Sosonkina, M. 1998. Solution of distributed sparse linear systems using psparslib. In *Applied Parallel Computing, Proc. PARA'98*, 501-509. Berlin: Springer-Verlag.

Salvini, S., Smith, B., and Greenfield, J. 1998. Towards mixed mode parallelism on the new model F50-based IBM SP system. Albuquerque HPCC Report AHPCC98-003.

Salvini, S., and Waniewski, J. 1996. Linear algebra subprograms on shared memory computers: Beyond LAPACK. In *Applied Parallel Computing Industrial Computation and Optimization, Third International Workshop, PARA'96*, edited by J. Waniewski, J. Dongarra, K. Madsen, and D. Olesen. Berlin: Springer-Verlag.

The SMP Library User Manual, Release 1. 1997. Oxford, UK: Numerical Algorithms Group Ltd.

Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J. 1996. *MPI: The Complete Reference*. Cambridge, MA: MIT Press.

Weerawarana, S., Houstis, E., Rice, R., Catlin, A., Crabill, C., and Chui, C. 1994. Pdelab, an object-oriented framework for building problem solving environments for PDE-based applications. Technical Report CSD-TR-94-021, Purdue University [Online]. Available: http://ziggurat.ca.sandia.gov/isis and http://www.cs.purdue.edu/research/cse/pellpack/pellpack.html.