Numerical Libraries and Tools for Scalable Parallel Cluster Computing

Shirley Browne†, Jack Dongarra†‡, and Anne Trefethen*

† University of Tennessee
‡ Oak Ridge National Laboratory
* Numerical Algorithms Group Ltd.

Introduction

For cluster computing to be effective within the scientific community it is essential that there are numerical libraries and programming tools available to application developers. Cluster computing may mean a cluster of heterogeneous components or hybrid architecture with some SMP nodes. This is clear at the high end, for example 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.

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, MPI-based libraries and tools will operate on cluster systems, but they 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; as 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 [0]. 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 [2] 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.

Numerical Libraries

The last 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. 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.) These 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 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. Since 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. 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 (SPMD) parallelism with message passing communication (e.g., MPI); Constellations 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 stand for Intelligent-RAM, and is also called 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 devices 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 modelling in previous work, but the complexity of Constellations and other architectures presents new challenges in this regard.

Another architecturally driven algorithmic opportunity arises on Pentium processors, which are not only widely used on desktops, but 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. There are a variety of algorithms that 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 like matrix-matrix multiplication, matrix-vector multiplication, FFTs, etc. 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 produced matrix multiply and FFT routines that are usually faster than the hand tuned codes from IBM, SGI and some other manufacturers. (See: http://www.netlib.org/atlas/ and http://www.fftw.org/)

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 [3, 19, 20] 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 (http://www.netlib.org) or on the National HPCC Software Exchange, NHSE (http://www.nhse.org).

As one might expect, the majority of available packages in this area are in linear algebra. Both direct solvers and iterative are well represented. Direct solver packages include ScaLAPACK (http://www.netlib.org/scalapack) [4] and PLAPACK (http://www.cs.utexas.edu/users/rvdg/plapack) [5] both of which are based on the parallelization of LAPACK [6] but by different approaches.

Iterative solver packages include Aztec (http://www.cs.sandia.gov/CRF/aztec1.html) [7], Blocksolve (http://www-unix.mcs.anl.gov/sumaa3d/BlockSolve) [8] and also PSPARSLIB (http://www.cs.umn.edu/Research/arpa/p_sparslib/psp-abs.html) [9]. 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 (http://www.caam.rice.edu/software/ARPACK) [10] and PeIGS [11]; 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 [12] based on MPI and also an SMP library [13,14] based on OpenMP.

The libraries that have been mentioned so far all have a traditional library interface. One of the packages that offer parallel functionality in the setting of an environment is PETSc [15,16]. 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 user develop their application in the PETSc style rather than calling a particular routine from PETSc. This provides an object-oriented interface which of course is also the natural interface for C++ -based packages such as ISIS++ (http://ziggurat.ca.sandia.gov/isis) and \\ELLPACK (http://www.cs.purdue.edu/research/cse/pellpack/pellpack.html) [17].

It is clear that there is a wealth of software available for clusters; however, as noted above this 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.

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 [18]. 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 but will be adapted have appropriate SMP-based kernels. This is likely to provide the natural progression to general numerical libraries for clustered computing.

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 interest for scientific computing on PC clusters include Fortran, C, and C++. Below is 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, for example with different compilers and MPI implementations.

Compilers and Preprocessors

The Absoft Pro Fortran toolset for Windows 95/98/NT/2000 includes globally optimizing Fortran 77, Fortran 90, and C/C++ compilers integrated with a development environment designed for Fortran users. Pro Fortran 6.2 is a Fortran toolset optimized for single processor Windows95/98 and Windows NT/2000 systems. Pro FortranMP 6.2 includes in addition a thread-safe runtime library and the VAST-F/Parallel preprocessor for automatically restructuring Fortran code for execution on dual processor systems. The Fx source-level debugger supporting Fortran and C/C++ is included with both toolsets, as is a performance profiler. All compilers and tools can be accessed via an integrated development environment (IDE) that automatically builds make files and generates header dependencies. The Absoft compilers are also available for Linux.

Compaq Visual Fortran (formerly Digital Visual Fortran) is available for Windows 95/98 and Windows NT along with a development environment.

Visual KAP is a preprocessor that automatically parallelizes Fortran 77/90/95 and ANSI C source code. Visual KAP runs on PentiumII/Pentium Pro/Pentium based machines under Windows NT 4.0 or Windows 95, and targets the Compaq Visual Fortran compiler. Since Windows 95 does not support parallelism, parallelism optimizations are available under Windows NT. Visual KAP

works with optimizing compilers to provide additional speedups beyond what the compiler's built-in optimizer provides. Visual KAP has both command-line and graphical user interfaces.

Microsoft Visual C++ is part of the Microsoft integrated development environment (IDE) called Visual Studio 6.0. The Visual C++ compiler can process both C source code and C++ source code. The compiler is compliant with all ANSI standards and has additional Microsoft extensions. The C++ Standard Template Library (STL) is included. The Visual C++ debugger has a graphical user interface for setting breakpoints, viewing classes and variables, etc. The debugger has an edit and continue feature that allows the user to change an application and continue debugging without manually exiting the debugger and recompiling.

VAST/Parallel from Pacific-Sierra Research Corporation includes the VAST-F/Parallel and VAST-C/Parallel automatic parallelizing preprocessors for Fortran and C, respectively. VAST/Parallel transforms and restructures the input source to use parallel threads so that the program can automatically make use of parallel processors. They also support the OpenMP standard for user-directed parallelization. VAST/Parallel can optionally produce a diagnostic listing that indicates areas for potential improvement to assist the developer in tuning performance further. VAST/Parallel works with the DEEP development environment by gathering compile-time data for DEEP and inserting instrumentation code for run-time data gathering. DEEP uses this information to display compile-time optimization notes (e.g., which loop nests have been parallelized, which data dependencies are preventing parallelization) and run-time performance data (e.g., which loop nests use the most wallclock time, which procedures are called the most.

MPI Implementations

MPICH is a freely available reference implementation of MPI developed by Argonne National Laboratory and Mississippi State University. Version 1.1.2 of MPICH is available for Linux on PCs and is in use on the Linux RoadRunner Supercluster at the University of New Mexico. Problems with MPICH programs on LINUX suddenly failing with lost TCP connections is on the list of things to fix in future MPICH releases. The MPICH development team at Argonne is involved in a research effort with LBNL to develop MPICH on top of VIA, the Virtual Interface Architecture, a specification of an industry-standard architecture for scalable communication within clusters. MPICH for Windows NT is expected to be released soon.

MPI-FM, developed by the Concurrent Systems Architecture Group at the University of Illinois at Urbana-Champaign and the University of California, San Diego, is a version of MPICH built on top of Fast Messages. MPI-FM requires the High Performance Virtual Machine (HPVM) runtime environment that is available for both Linux and Windows NT. MPI-FM is in use on the NCSA NT Supercluster.

MPI/Pro is a commercial MPI implementation from MPI Software Technology, Inc.. The current release of MPI/Pro is for Windows NT on Intel and Alpha processors, but MPI/Pro for Linux is expected to be released soon. MPI/Pro is based on a version of MPICH for Win32 platforms that was developed at Mississippi State. MPI Software Technology is working on a new version of MPI/Pro that does not include any MPICH code and that supports VIA.

PaTENT MPI 4.0 is a commercial MPI implementation for Windows NT available from Genias GmbH. PaTENT stands for Parallel Tools Environment for NT. Genias plans to relase a suite of development tools for MPI programs on NT, including debugging and performance analysis support.

Development Environments

Some vendors of Linux and Windows products are coming out with integrated development environments (IDEs) that integrate tools such as compilers, preprocessors, source code editors, debuggers, and performance analysis tools. Some IDEs are a complete product from a single vendor. Others provide a framework and some tools but integrate other tools, such as compilers, from other vendors.

DEEP from Pacific-Sierra Research stands for Development Environment for Parallel Programs. DEEP provides an integrated interactive GUI interface that binds performance, analysis, and debugging tools back to the original parallel source code. DEEP supports Fortran 77/90/95, C, and mixed Fortran and C in Unix and Windows 95/98/NT environments. DEEP supports both shared memory (automatic parallelization, OpenMP) and distributed memory (MPI, HPF, Data Parallel C) parallel program development. A special version of DEEP called DEEP/MPI is aimed at support of MPI programs.

Debuggers

DEEP/MPI from Pacific Sierra Research is a development environment for MPI parallel programs. DEEP/MPI debugging support is available for Linux on PCs. DEEP/MPI provides a graphical interface for parallel debugging of Fortran or C MPI programs. Capabilities include setting breakpoints, watching variables, array visualization, and monitoring process status. DEEP/MPI for Linux has been tested with MPICH and LAM MPI 6.1.

The PGDBG Debugger is part of the Portland Group, Inc. (PGI) PGI Workstation 3.0 development environment for Intel processor-based Linux, NT, and Solaris86 clusters. PGDBG supports threaded shared-memory parallelism for auto-parallelized and OpenMP programs, but does not provide explicit support for MPI parallel programming.

GDB, the GNU Debugger, is available for both Linux and Windows NT on PCs from Cygnus. GDB supports debugging of multithreaded code but does not provide explicit support for MPI parallel programming.

Performance Analyzers

DEEP/MPI from Pacific Sierra Research is a development environment for MPI parallel programs. DEEP/MPI performance analysis support is available for both Windows NT and Linux on PCs. DEEP/MPI provides a graphical user interface for program structure browsing, profiling analysis, and relating profiling results and MPI events to source code. DEEP/MPI for Linux has been tested with and supplies MPI profiling libraries for MPICH and LAM MPI 6.1. A driver called *mpiprof* is used to instrument, compile, and build MPI programs. Running *mpiprof* results in the creation of a subdirectory called *deep* in which the static information files created for each file you instrument are saved. When you execute your MPI program as you normally do, runtime information files are also stored in this subdirectory. To analyze following execution, start DEEP/MPI by executing the command *deep* which will bring up a File Dialog box asking you the location of your program. Currently DEEP requires that all source files be located in the same directory and that all static and runtime created files be located in the *deep* subdirectory, but this restriction is expected to be removed in future releases.

VAMPIR is a performance analysis tool for MPI parallel programs. The VAMPIRtrace MPI profiling library is available for Linux on PCs. VAMPIR provides a graphical user interface for analyzing tracefiles generated by VAMPIRtrace.

The Jumpshot graphical performance analysis tool is provided with the MPICH distribution. Jumpshot analyzes tracefiles produced by the MPE logging library, which is an MPI profiling library also provided with MPICH for Linux. Jumpshot is written in Java and interprets tracefiles in the binary clog format by displaying them onto a canvas object. Jimpshot itself is available for Windows (a JVM for Windows is provided with the Jumpshot distribution), and MPICH for Windows NT is supposed to be available soon. By default, Jumpshot shows a timeline view of the state changes and message passing behavior of the MPI processes. Clicking any mouse button on a specific state instance will display more information about that state instance. Clicking any mouse button on the circle at the origin of a message will display m ore information about that message. That Jumpshot's performance decreases as the size of the logfile increases is a known bug, and can ultimately result in Jumpshot hanging while it is reading in the logfile. There is a research effort underway to make jumpshot significantly more scalable. Jumpshot can be run as an application or as an applet using a web browser or applet viewer.

PGPROF is part of the Portland Group, Inc. (PGI) PGI Workstation 3.0 development environment for Intel processor-based Linux, NT, and Solaris86 clusters. PGPROF supports threaded shared-memory parallelism for auto-parallelized and OpenMP programs, but does not provide explicit support for MPI parallel programming. The PGPROF profiler supports function-level and line-level profiling of C, C++, Fortran 77, Fortran 90, and HPF programs. A graphical user interface displays and supports analysis of profiling results.

Paradyn is a tool for measuring and analyzing performance of large-scale long-running parallel and distributed programs. Paradyn operates on executable images by dynamically inserting instrumentation code while the program is running. Paradyn can instrument serial programs running on WindowsNT/x86 but does not yet support any MPI implementation on that platform.

VTune is a commercial performance analyzer for high-performance software developers on Intel processors, including Pentium III, under Windows NT. VTune collects, analyzes, and provides architecture-specific performance data from a system-wide view down to a specific module, function, or instruction in your code. VTune periodically interrupts the processor and collects samples of instruction addresses and matches these with application or operating system routines, and graphically displays the percentage of CPU time spent in each active module, process, and processor. VTune provides access to the Pentium hardware performance counters under Windows NT.

PAPI is a platform-independent interface to hardware performance counters. A PAPI implementation is available for Linux/x86.

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, however 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.

References

1. Snir, M., Otto, S. Huss-Lederman, S. Walker, D., and Dongarra, J., MPI: The Complete Reference, MIT Press, Boston, 1996.
2. Dagum, L., and Menon, R.,: OpenMP: An Industry-Standard API for Shared-Memory programming, in IEEE Computational Science & Engineering, Vol. 5, No. 1, January/March 1998.
3. Allan R.J, Hu Y.F., and Lockey P.: Parallel Application Software on High Performance Computers. Survey of Parallel Numerical Analysis Software, http://www.cse.clrc.ac.uk/Activity/HPCI
4. Blackford, L. S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I, Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Walker, D., Whaley, R.C.: ScaLAPACK Users' Guide, SIAM, Philadelphia, 1997.
5. Philip Alpatov, Greg Baker, Carter Edwards, John Gunnels, Greg Morrow, James Overfelt, Robert van de Geijn, Yuan-Jye J. Wu, "PLAPACK: Parallel Linear Algebra Libraries Design Overview" , SC97.
6. Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenny, A., Ostrouchov, S., and Sorenson, D.: Lapack Users' Guide, Release 2.0. SIAM, Philadelphia, 1995.
7. Hutchinson, S. A., Prevost, L. V., Tuminaro, R. S. and Shadid, J. N.: AZTEC Users' Guide: Version 2.0. Sandia National Labs, 1998.
8. Jones, M. T., and Plassman, P. E.: Blocksolve V1.1: Scalable library software for parallel solution of sparse linear systems. ANL Report 92/46, Argonne National Laboratory, December 1992.
9. Saad, Y and Sosonkina, M.: Solution of distributed sparse linear systems using psparslib. In Applied Parallel Computing: Proc. PARA'98, pages 501-9. Springer Verlag, Lecture Notes in Computer Science, Vol. 1541, 1998.
10. Lehoucq, R. B., Sorensen, D. C. and Yang, C.: ARPACK Users' Guide: Solution of Large-scale Eigenvalue Problems with implicitly-restarted Arnoldi Methods. SIAM, Philadelphia, 1998.
11. Elwood D., Fann, G., and Littlefield, R.: Parallel Eigensolver System User Manual. Batelle Pacific Northwest Laboratory. (Available from anonymous@ftp://pnl.gov)
12. The NAG Parallel Library Manual, Release 2, Numerical Algorithms Group Ltd, Oxford (1997).
13. Salvini, S., Waśniewski, J.: Linear Algebra Subprograms on Shared Memory Computers: Beyond LAPACK. In J.Waśniewski, J.Dongarra, K.Madsen, D.Olesen (eds.), Applied Parallel Computing Industrial Computation and Optimization, Third International Workshop, PARA'96, Lyngby, Denmark, August 18--21, 1996, Proceedings. Springer-Verlag Lecture Notes in Computer Science, Vol. 1184, 1996.
14. The SMP Library User Manual, Release 1, Numerical Algorithms Group Ltd, Oxford (1997).
15. Gropp W., Smith B.: Scalable, extensible, and portable numerical libraries. Technical report, Argonne National Laboratory.
16. Balay, S., Gropp W., Curfman McInnes, L. and Barry Smith: PETSc 2.0 Users' Manual. Argonne National Laboratory, 1996. Technical Report ANL-95/11 revision 2.0.17.
17. Weerawarana, S., Houstis, E. N. Rice R. J., Catlin, A. C., Crabill, C. L. and Chui, C. C. Pdelab: an object-oriented framework for building problem solving environments for PDE-based applications. Technical Report CSD-TR-94-021, Purdue University, March 1994.
18. Salvini, S., Smith, B., and Greenfield, J.: Towards Mixed Mode Parallelism on the New Model F50-based IBM SP System. Albuquerque HPCC report AHPCC98-003.

19. Eijkhout,V,: A Survey of Iterative Linear System Solver Packages, http://www.netlib.org/utk/papers/iterative-survey/
20. Dongarra, J.J., Freely Available Software For Linear Algebra On The Web, http://www.netlib.org/utk/people/JackDongarra/la-sw.html