Future Linear Algebra Libraries

Jack Dongarra

November 22, 1996

1. Traditional Libraries. The ultimate development of fully mature parallel scalable libraries will necessarily depend on breakthroughs in many other supporting technologies. Development of scalable libraries cannot wait, however, until all of the enabling technologies are in place. The reason is twofold: (1) the need for such libraries for existing and near-term parallel architectures is immediate, and (2) progress in all of the supporting technologies will be critically dependent on feedback from concurrent efforts in library development.

The linear algebra community has long recognized that we needed something to help us in developing our algorithms into software libraries. Several years ago, as a community effort, we put together a *de facto* standard for identifying basic operations required in our algorithms and software. Our hope was that the standard would be implemented on the machines by many manufacturers and that we would then be able to draw on the power of having that implementation in a rather portable way. We began with those BLAS operations designed for basic matrix computations. Since on a parallel system message passing is critical we have been involved with the development of message passing standards. Both PVM and MPI have helped in the establishment of standards and the promotion of portable software that is critical for software library work.

2. User Interfaces. As computer architectures and programming paradigms become increasingly complex, it becomes desirable to hide this complexity as much as possible from the end user. The traditional user interface for large, general-purpose mathematical and scientific libraries is to have users write their own programs (usually in Fortran) that call on library routines to solve specific subproblems that arise during the course of the computation. When extended to run on parallel architectures, this approach has only a limited ability to hide the underlying architectural and programming complexity from the user. As we extend the conventional notion of mathematical and scientific libraries to scalable architectures, we must rethink the conventional concept of user interface and devise alternate approaches that are capable of hiding architectural, algorithmic, and data complexity from users.

One possible approach is that of a "problem solving environment," typified by current packages like MATLAB, which would provide an interactive, graphical interface for specifying and solving scientific problems, with both algorithms and data structures hidden from the user because the package itself is responsible for storing and retrieving the problem data in an efficient distributed manner. Such an approach seems especially appropriate in keeping with the trend toward graphical workstations as the primary user access to computing facilities, together with networks of computational resources that include various parallel computers and conventional supercomputers. The ultimate hope would be to provide seamless access to such computational engines that would be invoked selectively for different parts of the user's computation according to whichever machine is most appropriate for a particular subproblem. We envision at least two interfaces for a library in linear algebra. One would be along conventional lines (LAPACK-style) for immediate use in conventional programs that are being ported to novel machines, and the other would be in the form of a problem solving environment (MATLAB-style). The two proposed interface styles are not inconsistent or incompatible: the problem solving environment can in fact be built on top of software that is based on a more conventional interface.

3. Heterogeneous Networking. Current trends in parallel architectures, high-speed networks, and personal workstations suggest that the computational environment of the future for working scientists will require the seamless integration of heterogeneous systems into a coherent problem-solving environment. Graphical workstations will provide the standard user interface, with a variety of computational engines and data storage devices distributed across a network. The diversity of parallel architectures means that inevitably different computational tasks will be more efficient on some than on others, with no single architecture uniformly superior. Thus, we expect the "problem-solving environment" envisioned above eventually to migrate to a heterogeneous network of workstations, file servers, and parallel computation servers. The various computational tasks required to solve a given problem would automatically and transparently be targeted to the most appropriate computational engine on the network. System resources would be shared among many users, but in a somewhat different manner than conventional timesharing computer systems. We have already made important first steps toward achieving these goals with systems like PVM and MPI, which supplies the low-level services necessary to coordinate the use of multiple workstations and other computers for individual jobs, and this system could serve as the foundation for a complete problem-solving environment of the type we envision.

Network computing techniques such as NetSolve offers the ability to look for computational resources on a network for a submitted problem (which can be a single LAPACK, ScaLAPACK or Matlab function call), choose the best one available, solve it (with retry for fault tolerance) and return the answer to the user. This system is available for Fortran, C, and Matlab users.

4. Software Tools and Standards. An ambitious development effort in scalable libraries will require a great deal of supporting infrastructure. Moreover, the portability of any library is critically dependent on adherence to standards. In the case of software for parallel architectures, precious few standards exist, so new standards must evolve along with the research and development. A particularly important area for scalable distributed-memory architectures is internode communication. The BLAS have proven to be very effective in assisting portable, efficient software for sequential and some of the current class of high-performance computers. We are investigating the possibility of expanding the set of standards that have been developed. There is a need for a light weight interface to much of the functionality of traditional BLAS. In addition, iterative and sparse direct methods require additional functionality not in traditional BLAS. Numerical methods for dense matrices on parallel computers require high efficiency kernels that provide functionality similar to that in traditional BLAS on sequential machines.

Software tools are also of great importance, both for developers to use in designing and tuning the library software, and for end-users to monitor the efficiency of their applications.

Conclusions

1. In spite of a lack of enabling technologies, library development cannot wait for research in programming languages, compilers, software tools, and other areas to mature, but must be done in conjunction with work in these areas. The the time to begin is now.

2. The user–library interface needs rethinking. It is not clear that the conventional library interface will be adequate to hide the underlying complexity from the user.

3. Object-oriented programming will be required to develop portable libraries that allow the user to work at an appropriate conceptual level.

4. Work on algorithms, particularly linear algebra, is important and cannot be isolated from general library development.

5. Language standards are important. The lack of language standards is the most significant obstacle to the development of communication libraries. A language standard must emerge before a software tool "development sweep" can begin.

These are some of the major research issues in developing scalable parallel linear algebra libraries.