ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers - Design Issues and Performance *

J. Choi[1], J. Demmel[2], I. Dhillon[1], J. Dongarra[1,3], S. Ostrouchov[1], A. Petitet[1], K. Stanley[1], D. Walker[3] and R.C. Whaley[1]

[1] Department of Computer Science, University of Tennessee
Knoxville, TN 37996-1301, USA
[2] Computer Science Division, University of California
Berkeley, CA 94720, USA
[3] Mathematical Sciences Section, Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA

Abstract. This paper outlines the content and performance of ScaLAPACK, a collection of mathematical software for linear algebra computations on distributed memory computers. The importance of developing standards for computational and message passing interfaces is discussed. We present the different components and building blocks of ScaLAPACK. This paper outlines the difficulties inherent in producing correct codes for networks of heterogeneous processors. Finally, this paper briefly describes future directions for the ScaLAPACK library and concludes by suggesting alternative approaches to mathematical libraries, explaining how ScaLAPACK could be integrated into efficient and user-friendly distributed systems.

1 Overview and Motivation

ScaLAPACK is a library of high performance linear algebra routines for distributed memory MIMD computers. It is a continuation of the LAPACK project, which designed and produced analogous software for workstations, vector supercomputers, and shared memory parallel computers. Both libraries contain routines for solving systems of linear equations, least squares problems, and eigenvalue problems. The goals of both projects are efficiency (to run as fast as possible), scalability (as the problem size and number of processors grow), reliability (including error bounds), portability (across all important parallel machines), flexibility (so users can construct new routines from well-designed parts), and ease-of-use (by making LAPACK and ScaLAPACK look as similar

* This work was supported in part by the National Science Foundation Grant No. ASC-9005933; by the Defense Advanced Research Projects Agency under contract DAAL03-91-C-0047, administered by the Army Research Office; by the Office of Scientific Computing, U.S. Department of Energy, under Contract DE-AC05-84OR21400; and by the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615.

as possible). Many of these goals, particularly portability, are aided by developing and promoting *standards*, especially for low-level communication and computation routines. We have been successful in attaining these goals, limiting most machine dependencies to two standard libraries called the BLAS, or Basic Linear Algebra Subroutines [5, 6, 12, 14], and BLACS, or Basic Linear Algebra Communication Subroutines [7, 9]. LAPACK and ScaLAPACK will run on any machine where the BLAS and the BLACS are available.

This paper presents the design of ScaLAPACK. After a brief discussion of the BLAS and LAPACK, the block cyclic data layout, the BLACS, the PBLAS (Parallel BLAS), and the algorithms used are discussed. We also outline the difficulties encountered in producing correct code for networks of heterogeneous processors; difficulties we believe are little recognized by other practitioners.

Finally, the paper discusses the performance of ScaLAPACK. Extensive results on various platforms are presented. One of our goals is to model and predict the performance of each routine as a function of a few problem and machine parameters. One interesting result is that for some algorithms, speed is *not* a monotonic increasing function of the number of processors. In other words, speed can be increased by letting some processors remain idle.

2 Design of ScaLAPACK

2.1 Portability, Scalability and Standards

In order to be truly portable, the building blocks underlying parallel software libraries must be *standardized*. The definition of computational and message-passing standards [10, 12] provides vendors with a clearly defined base set of routines that they can optimize. From the user's point of view, standards ensure portability. As new machines are developed, they may simply be added to the network, supplying cycles as appropriate.

From the mathematical software developer's point of view, portability may require significant effort. Standards permit the effort of developing and maintaining bodies of mathematical software to be leveraged over as many different computer systems as possible. Given the diversity of parallel architectures, portability is attainable to only a limited degree, but machine dependences can at least be isolated.

Scalability demands that a program be reasonably effective over a wide range of numbers of processors. The scalability of parallel algorithms over a range of architectures and numbers of processors requires that the granularity of computation be adjustable. To accomplish this, we use block algorithms with adjustable block sizes. Eventually, however, polyalgorithms (where the actual algorithm is selected at runtime depending on input data and machine parameters) may be required.

Scalable parallel architectures of the future are likely to use physically distributed memory. In the longer term, progress in hardware development, operating systems, languages, compilers, and communication systems may make it