Half Day Tutorial on

Using and Abusing PLAPACK: Parallel Linear Algebra Package

Submitted to SC98

Instructors

Greg Morrow
Robert van de Geijn
Department of Computer Sciences
and
Texas Institute for Computational and Applied Mathematics
The University of Texas at Austin Austin, TX 78712
{morrow,rvdg}@cs.utexas.edu

Corresponding instructor:

Robert van de Geijn

Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
(512) 471-9720 (O)
(512) 471-8885 (F)
rvdg@cs.utexas.edu
http://www.cs.utexas.edu/users/rvdg

Abstract

PLAPACK is a new parallel (dense) linear algebra infrastructure designed to correct fundamental problems that have surfaced for the last fifteen years when designing and implementing parallel dense linear algebra algorithms and libraries:

Interface: Traditional approaches require applications to build matrices and vectors using a class of cartesian matrix distributions. Since matrices are often created "additively", this puts a considerable burden on applications, which must manage communication required or, alternative, perform redundant computation. PLAPACK solves this problem by providing a convenient interface that hides all communication necessary to fill and query matrices and vectors.

Data distribution: Traditional approaches concentrate on the distribution of matrices, and treats vectors as matrices of width one. The problem with this is that problem domains are more closely correlated with vectors. Since matrices are distributed using 2D cartesian distributions, vectors are assigned to one column of processors, and thus the problem domain is expected to be assigned to one column of processors. PLAPACK solves this problem by distribution of vectors equally to all processors, and *inducing* a 2D cartesian matrix distribution from this vector distribution.

Abstraction: Traditional approaches code algorithms at a level of abstraction similar to that used for classic sequential libraries. The problem with this is that indices are explicitly exposed, and, more importantly, to achieve high performance, manual translation from indexing at a global level to indexing at the processor level must be performed. This is tedious, error prone, and results in code size explosion. PLAPACK solves this problem by using an object based (MPI-like) programming style, which hides *all indexing* into individual elements of matrices and vectors, both at the global and local level. The resulting coding style allows for direct translation of algorithms as they are naturally explained in a classroom setting to highly efficient code. It is this feature that makes PLAPACK attractive both as an infrastructure for library development as well as for education.

PLAPACK achieves these goals without sacrificing high performance or portability: PLAPACK Cholesky factorization attains more than 350 MFLOPS per processing element on the Cray T3E-600 and PLAPACK has been ported to the Cray T3E, IBM SP-2, Convex SPP-2000, "Beowulf" systems based on Pentium processors (both for Linux and NT), SGI PowerChallenge, networks of alpha workstations, and other distributed memory systems. Additional features of PLAPACK include transparent support for out-of-core capabilities and interfaces to math environments like Mathematica, MATLAB, HiQ, and LabView.

Intended Audience

Library users, library developers, and educators

Beginner – 25%
Intermediate – 50%
Advanced – 25%

The tutorial assumes that participants have familiarity with MPI and basic dense linear algebra algorithms.

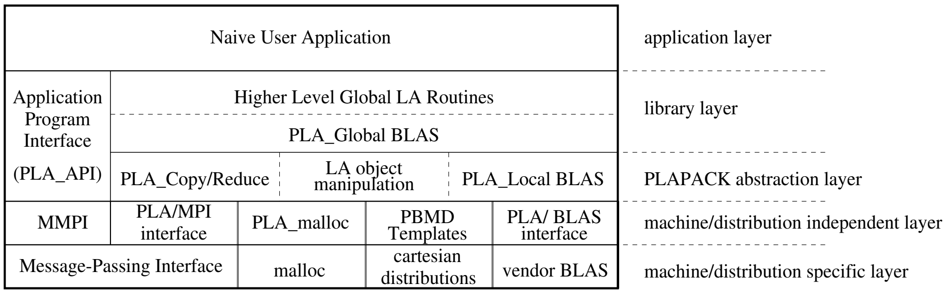
Equipment

Although not finalized, Intel has shown an interest in providing a "traveling Beowulf" system for this tutorial, in form of a network of 8–16 Pentium based laptops connected with a fast switch.

1 Description of Tutorial Material

1.1 Introduction (15 min)

In the introduction, we describe why traditional approaches to parallel dense linear algebra libraries have fallen short. We motivate a new philosophy, which places the application (i.e. the *vectors* in a linear system) at the center of the distribution of vectors and matrices to processors. We show the benefits of encoding information about vectors and matrices in *opaque* objects in an MPI-like programming style. Finally, we motivate the layering in the PLAPACK infrastructure, as presented in Fig. 1.



O

Figure 1: Layering of the PLAPACK infrastructure

1.2 Interfacing an application with PLAPACK (45 min)

In this part of the tutorial, we show the participants how to create and manipulate distributed matrices and vectors, how to fill them in an application-friendly fashion with data, and how to call a simple PLAPACK library routine. Finally, we show how to extract information from the distributed matrices and vectors. An example of a simple fill routine in given in Fig. 2. We will present the interface for both C and Fortran. Upon completion of this part of the tutorial, we will have developed an example code that can be used to interface real applications with PLAPACK.

1.3 Parallelizing a dense linear algebra algorithm using PLAPACK (30 min)

In this part of the tutorial, we show how blocked algorithms can be developed that achieve high performance using Level-1, 2, and 3 Basic Linear Algebra Subprograms (BLAS). Next, we show how to encode such algorithms using the PLAPACK infrastructure, where each line of the resulting code is a direct translation of the algorithm *without the introduction of indices or loop counters*. Let us illustrate this for the Cholesky factorization:

1.3.1 Planning the algorithm

The Cholesky factorization of a square matrix A is given by

$$A \rightarrow LL^T$$

where A is symmetric positive definite and L is lower triangular.

The algorithm for implementing this operation can be described by partitioning the matrices

$$A = \left(\begin{array}{c|c} A_{11} & \star \\ \hline A_{21} & A_{22} \end{array} \right) \quad \text{and} \quad L = \left(\begin{array}{c|c} L_{11} & 0 \\ \hline L_{21} & L_{22} \end{array} \right)$$

```
#include "mpi.h"
#include "PLA.h"

void create_problem( PLA_Obj A, PLA_Obj x, PLA_Obj b )
{
  PLA_Obj  zero = NULL,    one = NULL;
  int size, me, nprocs, i, j;
  double *locA, d_one = 1.0;

  PLA_Obj_global_length( A, &size );

  MPI_Comm_rank( MPI_COMM_WORLD, &me );
  MPI_Comm_size( MPI_COMM_WORLD, &nprocs );

  PLA_Create_constants_conf_to( A, NULL, &zero, &one );

  PLA_Obj_set_to_zero ( A );        PLA_Obj_set_to_zero ( x );
  PLA_Obj_set_to_zero ( b );

  PLA_API_begin();
    PLA_Obj_API_open(A);            PLA_Obj_API_open(x);

    srand48( me * 1793 );

    locA = (double *) malloc( sizeof( double ) * size  );
    for (j=me;j< size; j+=nprocs) {
      for (i=0; i < size; i++) locA[i]=drand48() * 2.0 - 1.0;
      locA[j] += size;
      PLA_API_axpy_matrix_to_global( size, 1, &d_one, locA, size, A, 0, j );
    }

    if (0==me) {
      for (i=0; i<size; i++) locA[i]=drand48() * 2.0 - 1.0;
      PLA_API_axpy_vector_to_global( size, &d_one, locA, 1, x, 0);
    }
    free( locA );

    PLA_Obj_API_close(x);           PLA_Obj_API_close(A);
  PLA_API_end();

  PLA_Symv( PLA_LOW_TRIAN, one, A, x, zero, b );

  PLA_Obj_free( &zero );         PLA_Obj_free( &one );
}
```

Figure 2: Filling a global matrix by columns.

where A_{11} and L_{11} are $b \times b$ ($b << n$). The $\star$ indicates the symmetric part of A. Now,

$$A = \left(\begin{array}{c|c} A_{11} & \star \\ \hline A_{21} & A_{22} \end{array} \right) = \left(\begin{array}{c|c} L_{11} & 0 \\ \hline L_{21} & L_{22} \end{array} \right) \left(\begin{array}{c|c} L_{11}^T & L_{21}^T \\ \hline 0 & L_{22}^T \end{array} \right) = \left(\begin{array}{c|c} L_{11}L_{11}^T & \star \\ \hline L_{21}L_{11}^T & L_{21}L_{21}^T + L_{22}L_{22}^T \end{array} \right)$$

This in turn yields the equations

$$\begin{array}{rcl} A_{11} & \rightarrow & L_{11}L_{11}^T \\ L_{21} & = & A_{21}L_{11}^{-T} \\ A_{22} - L_{21}L_{21}^T & \rightarrow & L_{22}L_{22}^T \end{array}$$

The above motivation yields an algorithm as it might appear on a chalk board in a classroom, in Figure 3 (a). In that figure, we take $b = 1$, so that A_{11} is a scalar and A_{21} is a vector. Since A is symmetric, only the lower triangular part of the matrix is stored and only the lower triangular part is updated.

1.3.2 Translation to PLAPACK

The PLAPACK implementation of the above described algorithm is given in Figure 3 (b):

let $A_{\rm cur} = A$

do until $A_{\rm cur}$ is 0×0

Partition

$$A_{\rm cur} = \left(\begin{array}{c|c} a_{11} & \star \\ \hline a_{21} & A_{22} \end{array} \right)$$

$a_{11} \leftarrow l_{11} = \sqrt{a_{11}}$

$a_{21} \leftarrow l_{21} = a_{21}/l_{11}$

$A_{22} \leftarrow A_{22} - l_{21}l_{21}^T$

continue with $A_{\rm cur} = A_{22}$

(a)

```
PLA_Obj_view_all( a, &acur );
while ( TRUE ){
     PLA_Obj_global_length( acur, &size );
     if ( 0 == size ) break;
     PLA_Obj_split_4( acur, 1, 1, &a11, &a12,
                                   &a21, &acur );
     Take_sqrt( a11 );
     PLA_Inv_scal( a11, a21 );
     PLA_Syr( PLA_LOW_TRIAN, min_one, a21, acur );
}
```

(b)

Figure 3: Algorithm for computing the Cholesky factorization of a matrix and its PLAPACK implementation

All information that describes A, its data, and how it is distributed to processors (nodes) is encoded in a object (data-structure) referenced by `a`. The `PLA_Obj_view_all` creates a second reference, `acur`, into the same data. The `PLA_Obj_global_length` call extracts the current size of the matrix that `acur` currently references. If this is zero, the recursion has completed. The call to `PLA_Obj_split_4` partitions the matrix, creating new references for the four quadrants. Element a_{11} is updated by taking its square root in subroutine `Take_sqrt`, a_{21} is scaled by $1/a_{11}$ by calling `PLA_Inv_scal`, and finally the symmetric rank-1 update is achieved through the call to `PLA_Syr`. This sets up the next iteration, which is also the next level of recursion in our original algorithm.

1.4 Optimization (30 min)

In this part of the tutorial, we further develop the example of the Cholesky factorization, showing how further optimizations can be achieved by explicitly indicating duplication of data, communicating without exposing actual communication calls, and managing local communication. While this allows one to achieve near-peak performance, still all indexing is hidden from the user, as is communication. In Fig. 4, we show a number of such optimizations. The final implementation presented is an algorithm which would be very difficult to implement without the PLAPACK infrastructure. Representative performance is given in Fig. 5

1.5 Customizing PLAPACK Code (45 min)

While the previous part of the tutorial introduces various techniques for optimizing PLAPACK implementations, we now concentrate on the ease with which existing implementations can be customized for a specific need. To demonstrate this, we will derive a high performance implementation of the QR factorization, and subsequently show how by changing a few lines of the implementation, it can be customized to perform "updating" of the factorization. This customization makes the QR factorization more appropriate for use in signal processing when signals arrive incrementally rather than being available *a priori*.

1.6 Conclusion (15 min)

We conclude the tutorial by summarizing extensions of PLAPACK that include MATLAB/Mathematica/HiQ/Labview interfaces and use of PLAPACK for iterative methods.

2 More Information on PLAPACK

See

`http://www.cs.utexas.edu/users/plapack`

3 Sample Materials

- PLAPACK Users' Guide: *Using PLAPACK: Parallel Linear Algebra Package*, The MIT Press, Spring 1997. An on-line version of this book is available via the PLAPACK webpage.
- We are developing an on-line tutorial from which materials will be taken for this course. This tutorial can be accessed via the PLAPACK webpage.

4 About the Instructors

Robert van de Geijn is an Associate Professor of Computer Sciences at the University of Texas at Austin. After obtaining a Ph.D. from the University of Maryland, he joined the University of Texas at Austin in 1987. He spend the 1990-91 academic year at the University of Tennessee where he became involved in the ScaLAPACK project. His interests are in Numerical Analysis, Parallel Numerical Algorithms and Parallel Communication Algorithms.

Greg Morrow received his Ph.D. in Physics from The University of Texas at Austin in Spring 1997. Since then, he has been a postdoctoral research scientist in computer sciences involved in the PLAPACK project. His primary research interests include Stochastic and Quantum Chaos, Parallel Numerical Algorithms, and Parallel Software Engineering.

partition $A = \left(\begin{array}{c|c} A_{TL} & \star \\ \hline A_{BL} & A_{BR} \end{array} \right)$

where A_{TL} is 0×0

do until A_{BR} is 0×0

determine block size b

repartition

$$\left(\begin{array}{c||c} A_{TL} & \star \\ \hline\hline A_{BL} & A_{BR} \end{array} \right) = \left(\begin{array}{c||c|c} A_{00} & \star & \star \\ \hline\hline A_{10} & A_{11} & \star \\ \hline A_{20} & A_{21} & A_{22} \end{array} \right)$$

where A_{11} is $b \times b$

$A_{11} \leftarrow L_{11} = \text{Chol. fact.}(A_{11})$

$A_{21} \leftarrow L_{21} = A_{21} L_{11}^{-T}$

$A_{22} \leftarrow A_{22} - L_{21} L_{21}^{T}$

continue with

$$\left(\begin{array}{c||c} A_{TL} & \star \\ \hline\hline A_{BL} & A_{BR} \end{array} \right) = \left(\begin{array}{c|c||c} A_{00} & \star & \star \\ \hline A_{10} & A_{11} & \star \\ \hline\hline A_{20} & A_{21} & A_{22} \end{array} \right)$$

enddo

(a) Blocked algorithm

```
PLA_Obj_view_all( A, &ABR );
while ( TRUE ) {
  PLA_Obj_global_length( ABR , &b );
  b = min( b, nb );
  if ( 0 == b ) break;

  PLA_Obj_split_4( ABR, b, b, &A11, &A12,
                              &A21, &ABR );

  Chol_level2( A11 );

  PLA_Trsm( PLA_SIDE_RIGHT, PLA_LOW_TRIAN,
            PLA_TRANS, PLA_NONUNIT_DIAG,
            one, A11, A21 );
  PLA_Syrk( PLA_LOW_TRIAN, PLA_NO_TRANS,
                minus_one, A21, one, ABR );
}
```

(b) Basic PLAPACK implementation

```
PLA_Obj_view_all( A, &ABR );
while ( TRUE ) {
  PLA_Obj_split_size( ABR, PLA_SIDE_LEFT,
                      &b_l, &owner_l );
  PLA_Obj_split_size( ABR, PLA_SIDE_TOP,
                      &b_t, &owner_t );
  b = min( min( b_t, b_l ), nb );
  if ( 0 == b ) break;

  PLA_Obj_split_4( ABR, b, b, &A11, &A12,
                              &A21, &ABR );
  PLA_Obj_objtype_cast( A11, PLA_MSCALAR );

  PLA_Local_chol( A11 );

  PLA_Mscalar_create_conf_to( A11,
        PLA_ALL_ROWS, PLA_INHERIT, &A11_dup );
  PLA_Copy( A11, A11_dup );

  PLA_Local_trsm( PLA_SIDE_RIGHT, PLA_LOW_TRIAN,
                  PLA_TRANS, PLA_NONUNIT_DIAG,
                  one, A11_dup, A21 );
  PLA_Syrk( PLA_LOW_TRIAN, PLA_NO_TRANS,
            minus_one, A21, one, ABR );
}
```

(c) Basic optimizations

```
PLA_Obj_split_4( A, 0, 0, &ATL, &ATR,
                          &ABL, &ABR );
PLA_Mvector_create_conf_to( ABL, nb, &A1_mv );

while ( TRUE ) {
  PLA_Obj_global_length( ABR, &b );
  b = min( b, nb );
  if ( 0 == b ) break;

  PLA_Obj_vert_split_2( ABR,   b, &A1,     &A2 );
  PLA_Obj_vert_split_2( A1_mv, b, &A1_mv1, &A1_mv2 );

  PLA_Copy( A1, A1_mv1 );
  PLA_Chol_mv( A1_mv1 );
  PLA_Copy( A1_mv1, A1 );

  PLA_Obj_split_4( ABR, b, b, &A11, &A12,
                              &A21, &ABR );
  PLA_Syrk( PLA_LOW_TRIAN, PLA_NO_TRANS,
            minus_one, A21, one, ABR );

  PLA_Obj_horz_split_2( A1_mv, b, PLA_DUMMY,
                                  &A1_mv );
}
```

(d) New parallel algorithm

Figure 4: Various versions of level-3 BLAS based Cholesky factorization using PLAPACK.

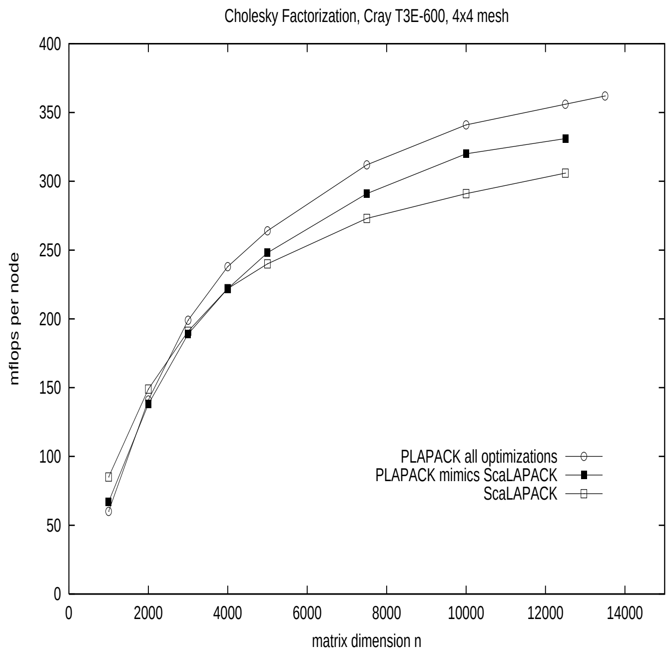


Figure 5: Performance of the various algorithms on a 16 processor configuration of the Cray T3E 600.

5 Related work by the instructors

5.1 Related Books

- Robert A. van de Geijn, *Using PLAPACK*, The MIT Press, 1997

5.2 Related Journal Papers

- Jack Dongarra, Robert van de Geijn and David Walker, “Scalability Issues Affecting the Design of a Dense Linear Algebra Library,” Special Issue on Scalability of Parallel Algorithms, *Journal of Parallel and Distributed Computing*, Vol. 22, No. 3, Sept. 1994.
- Tom Cwik, Robert van de Geijn, and Jean Patterson, “The Application of Parallel Computation to Integral Equation Models of Electromagnetic Scattering,” *Journal of the Optical Society of America A*, Vol. 11, No. 4, pp. 1538–1545, April 1994
- Jack Dongarra and Robert van de Geijn, “Reduction to Condensed Form on Distributed Memory Architectures,” in *Parallel Computing*, 18, pp. 973–982, 1992.
- Po Geng, J. Tinsley Oden, and Robert van de Geijn, “Massively Parallel Computation for Acoustical Scattering Problems using Boundary Element Methods,” *Journal of Sound and Vibration,* 191 (1), pp. 145-165.
- Almadena Chtchelkanova, John Gunnels, Greg Morrow, James Overfelt, and Robert van de Geijn, “Parallel Implementation of BLAS: General Techniques for Level 3 BLAS,” *Concurrency: Practice and Experience* to appear.

- Brian Grayson and Robert van de Geijn, "A High Performance Parallel Strassen Implementation," *Parallel Processing Letters*, Vol 6, No. 1 (1996) 3-12.
- Robert van de Geijn and Jerrell Watts, "SUMMA: Scalable Universal Matrix Multiplication Algorithm," *Concurrency: Practice and Experience*, to appear.

5.3 Related Conference Publications

- Greg Baker, John Gunnels, Greg Morrow, Beatrice Riviere, and Robert van de Geijn, "PLAPACK: High Performance through High Level Abstraction," Proceedings of ICPP98, to appear.
- John Gunnels, Calvin Lin, Greg Morrow, and Robert van de Geijn, "A Flexible Class of Parallel Matrix Multiplication Algorithms", Proceedings of IPPS98, 1998.
- Philip Alpatov, Greg Baker, Carter Edwards, John Gunnels, Greg Morrow, James Overfelt, Robert van de Geijn, Yuan-Jye J. Wu, "PLAPACK: Parallel Linear Algebra Package – Design Overview" , Proceedings of SC97, 1997.
- Philip Alpatov, Greg Baker, Carter Edwards, John Gunnels, Greg Morrow, James Overfelt, Robert van de Geijn, "PLAPACK: Parallel Linear Algebra Package," in Proceedings of the SIAM Parallel Processing Conference, 1997.
- Wu, Y.-J. J., A. A. Alpatov, C. Bischof, and R. A. van de Geijn, "A Parallel Implementation of Symmetric Band Reduction Using PLAPACK", Proceedings of Scalable Parallel Library Conference, Mississippi State. (October 1996, 8 pages, PRISM WN#35)
- Ken Klimkowski and Robert van de Geijn, "Anatomy of an out-of-core dense linear solver", *Proceedings of the 1995 International Conference on Parallel Processing,* pp. 29–33.
- J. G. Lewis, D. G. Payne, and R. A. van de Geijn, "Matrix-Vector Multiplication and Conjugate Gradient Algorithms on Distributed Memory Computers," *Scalable High Performance Computing Conference 1994*.
- J. G. Lewis and R. A. van de Geijn, "Implementing Matrix-Vector Multiplication and Conjugate Gradient Algorithms on Distributed Memory Multicomputers," Supercomputing '93.
- James Demmel, Jack Dongarra, Robert van de Geijn and David Walker, "LAPACK for Distributed Memory Architectures: The Next Generation," in *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing,* Norfolk, March 1993.
- Jack Dongarra, Robert van de Geijn, and David Walker), "A Look at Scalable Dense Linear Algebra Libraries," in *Proceedings of Scalable High Performance Concurrent Computing '92 (SHPCC92)*, April 27-29, 1992.
- Anderson, E., Benzoni, A., Dongarra, J., Moulton, S., Ostrouchov, S., Tourancheau, B., and van de Geijn, R., "Basic Linear Algebra Communication Subprograms." in *Sixth Distributed Memory Computing Conference Proceedings*, IEEE Computer Society Press, 1991, pp. 287–290.
- Anderson, E., Benzoni, A., Dongarra, J., Moulton, S., Ostrouchov, S., Tourancheau, B., and van de Geijn, R., "LAPACK for Distributed Memory Architectures: Progress Report." in *the Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, Philadelphia, 1992, pp. 625–630.

6 References

G.W. Stewart
Department of Computer Science
University of Maryland
College Park, MD 20742
stewart@cs.umd.edu

Steven Huss-Lederman
Department of Computer Science
University of Wisconsin–Madison
1210 W. Dayton St.
Madison, WI 53706
lederman@cs.wisc.edu

Chris Bischof
Mathematics and Computer Science Division
Argonne National Laboratory
9700 S. Cass Ave.
Argonne, IL 60439
bischof@mcs.anl.gov

6.1 Preface from *Using PLAPACK: Parallel Linear Algebra Package*, The MIT Press, 1997

The development of the Parallel Linear Algebra Package (PLAPACK, pronounced PLAY-pack) infrastructure resulted in part from my frustration while teaching a graduate level special topics course titled *Parallel Techniques for Numerical Algorithms* at the University of Texas. While I could explain the high performance scalable implementation of algorithms like matrix-matrix multiplication and Cholesky factorization without filling up more than half a chalk board, presenting actual parallel code for such operations required explanation of subroutine calls with fifteen to thirty parameters. The natural blocking that can be used to describe the algorithm, and is required to attain high performance, did not translate well to code.

To get a sense of how linear algebra algorithms can be naturally expressed as blocked algorithms, we suggest the reader now turn to the description of the Cholesky factorization algorithm in Section **??**. One will observe that the description of the algorithm does not index individual elements explicitly. There is an inherent use of recursion. The parallel algorithm is described without explicitly referencing processor indices or communication. Thus, an ideal infrastructure for coding such algorithms should capture these same attributes.

The arrival of the Message-Passing Interface (MPI) has not only provided an ideal vehicle for the communication layer of parallel libraries, but has also introduced a convenient approach to programming to the parallel processing community. One of the ways MPI reduces the number of parameters in the calling sequences of routines is through the use of *object based* programming. Elaborate descriptions of groups of processors within which an MPI communication is to occur are stored in a hidden data structure (opaque *object*). Similarly, descriptions of distributed matrices and vectors in PLAPACK are stored in *linear algebra* objects. Furthermore, we use *views*, objects that are references into distributed matrices and vectors, to address sub-blocks of matrices and vectors. Through use of views, a PLAPACK implementation becomes a line-by-line translation of a given blocked algorithm.

There are a number of additional recent advances that have been incorporated into PLAPACK. One is the recognition that applications inherently view the distribution of a matrix differently than has been traditionally supported by parallel linear algebra libraries. PLAPACK uses an alternative view of matrix distribution, *Physically Based Matrix Distribution*, in which the distribution of matrices is *induced* by the distribution of vectors in a linear system of equations. It is this approach to matrix distribution that also naturally supports and exploits the collective communications required to perform the data duplications necessary to parallelize dense linear algebra algorithms. This in turn supports a systematic, layered approach for the parallel implementation of common matrix-vector and matrix-matrix operations, allowing a highly compact implementation of the PLAPACK infrastructure.

PLAPACK can be used as a library. This requires a user to have a rudimentary knowledge of the first few chapters of this book and to access the PLAPACK web page, to see what higher level operations are available. It can be used as a vehicle for parallel implementation of new linear algebra algorithms, thus providing users with additional high level operations. In addition, its simplicity make it an ideal vehicle for education in the area of high performance supercomputing. Thus, we target scientific application programmers, library developers, and novices.

We intend for PLAPACK to become an "open" infrastructure, with new subroutines added as they are developed by our own group as well as others. For this purpose, we maintain an extensive web page at

`http://www.cs.utexas.edu/users/plapack/`

Since the name can be confusing, the PLAPACK library should not be mistaken for a parallelization of LAPACK [**?**] (generally pronounced L-A-pack) or a repackaging of ScaLAPACK [**?**] (generally pronounced Sca-L-A-pack). LAPACK is a highly successful high performance dense linear algebra package for conventional (shared memory) supercomputers as well as workstations. It was written in FORTRAN and makes use of the Level 1, 2, and 3 Basic Linear Algebra Subprograms (BLAS) to attain high performance in a portable fashion. We view ScaLAPACK as an effort to port LAPACK to distributed memory computers, with an emphasis on maximal code reuse through minimal change to all components of the LAPACK library. The PLAPACK approach to parallelizing linear algebra algorithms is somewhat more radical.

6.1.1 Acknowledgments

My first introduction to parallel computing came in the early eighties, when I was a Research Assistant for Prof. G. W. Stewart and Prof. Dianne O'Leary, at the University of Maryland in College Park. At the time, we were happy to have an experimental parallel computer, the *ZMOB*, consisting of 128 Z80 processors These processors were slow enough that we never did compute MFLOPS/sec attained. My introduction to *high performance* parallel computing occurred while I was on a leave of absence at the University of Tennessee. It was during this leave that Prof. Jack Dongarra introduced me to LAPACK and the distributed memory version of LAPACK that was being developed, which later evolved into ScaLAPACK.

Collaboration with Dr. David Payne, Lance Shuler and Jerrell Watts on the Interprocessor Collective Communication (InterCom) Project [**?**] showed how to systematically implement collective communication libraries for distributed memory architectures. Indeed, a large part of the introductory chapter of this book was taken from Part II of a manuscript authored with Payne, Shuler and Watts, with the working title "A Street guide to Collective Communication." The systematic approach used by PLAPACK to redistribute vectors and matrices was to be an illustration of the use of collective communication in that effort.

The fact that there was an inherent link between matrix distribution and vector distribution first became obvious to us in a paper authored with Dr. John Lewis and Dr. David Payne. In that paper [**?**], we discussed the efficient implementation of matrix-vector multiplication and showed how communication overhead is reduced by using a distribution that we now call Physically Based Matrix Distribution. It was the insights into applications provided by Carter Edwards, Dr. Abani Patra, and Dr. Po Geng that further provided evidence that this distribution provides a common distribution appropriate for dense, sparse iterative, and sparse direct linear algebra methods [**?**].

The systematic approach to parallel implementation of the level-3 Basic Linear Algebra Subprograms started with the Scalable Universal Matrix Multiplication Algorithm (SUMMA) [**?**], a collaboration with Jerrell Watts. This was later extended to all level-3 BLAS in a collaboration with Dr. Almadena Chtchelkanova, John Gunnels, Greg Morrow, and James Overfelt [**?**]. Interestingly enough, most of this work was inspired by looking at parallel out-of-core implementations of the LU factorization, first with Mercedes Marques and later with Ken Klimkowski [**?**].

We would like to thank Dr. Yuan-Jye (Jason) Wu for agreeing to be a one-man alpha-release test site. By implementing a complex algorithm (the reduction of a square dense matrix to banded form [**?**]) using PLAPACK, Jason exercised many components of the infrastructure, thereby revealing many initial bugs. We also appreciate the encouragement received from Dr. Chris Bischof, Dr. Steven Huss-Lederman, Prof. Xiaobai Sun, and others involved in the PRISM project. We gratefully acknowledge George Fann at PNNL for his support of the PLAPACK project and for his time collecting the IBM SP-2 performance numbers presented later in the guide. Similarly, we are indebted to Dr. Greg Astfalk for assembling the numbers for the Convex Exemplar.

It has been a great pleasure to work with the graduate students listed as contributors to the various chapters. As of this writing, they are all Ph.D. students at The University of Texas at Austin. Their fields of study span Aerospace Engineering (Greg Baker), Computational and Applied Mathematics (Carter Edwards and James Overfelt), Computer Science (John Gunnels), and Physics (Philip Alpatov and Greg Morrow). All started as students in my special topics course on Parallel Techniques for Numerical Algorithms. A number of them became Research Assistants as part of the project. Others volunteered their time (supported by other projects). It is through their talent and enthusiasm that the PLAPACK project progressed as rapidly as it has.

To date, the PLAPACK project has been supported in part by the following funding sources:

- The Parallel Research on Invariant Subspace Methods (PRISM) project, under ARPA grant P-95006.
- The NASA High Performance Computing and Communications Program's Earth and Space Sciences Project under NRA Grants NAG5-2497 and NAG5-2511.
- The Environmental Molecular Sciences construction project at Pacific Northwest National Laboratory (PNNL). PNNL is a multiprogram national laboratory operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract DE-AC06-76RLO 1830.

- The Intel Research Council.

We gratefully acknowledge access provided by the following parallel computing facilities:

- The University of Texas Computation Center's High Performance Computing Facility.
- The Texas Institute for Computational and Applied Mathematics' Distributed Computing Laboratory.
- The Molecular Science Computing Facility in the Environmental Molecular Sciences Laboratory at the Pacific Northwest National Laboratory (PNNL).
- The Intel Paragon system operated by the California Institute of Technology on behalf of the Concurrent Supercomputing Consortium. Access to this facility was arranged by the California Institute of Technology.
- Cray Research, a Silicon Graphics Company.
- The Convex Division of the Hewlett-Packard Company in Richardson, Texas.

7 Vita: Robert A. van de Geijn

Education

B.S. in Mathematics and Computer Science, University of Wisconsin-Madison, Aug. 1981. Ph.D. in Applied Mathematics, University of Maryland, College Park, Aug. 1987.

Academic Experience

Associate Professor of Computer Sciences, UT-Austin, Sept. 1994 - present.

Assistant Professor of Computer Sciences, UT-Austin, Sept. 1987 - Aug. 1994.

Research Professor of Computer Sciences, Univ. of TN - Knoxville, Aug. 1990 - May 1991

Awards

Dean's Fellow 1997-98 (Curriculum Development Award)

Faculty Fellowship #5 in Computer Sciences, (1995-present)

IBM Predoctoral Fellowship, 1986-87

Student Supervision

Ph.D. Students: Mike Barnett (CS, 1992, with Chris Lengauer), Timothy Collins (CS, 1995, with J.C. Browne), Carter Edwards (CAM, 1997, with L. Hayes and M. Wheeler), John Gunnels (CS, expected 1998), James Overfelt (CAM, expected 1998)

Master's Students: James Juszczak (CS, 1989), Lance Shuler (Math, Fall 1992), Martin Hoff (Math, 1993), Kenneth Klimkowski (CAM, 1997)

Postdoctoral: Greg Morrow, Jurgen Singer, Anna Elster

Related Books

- Robert A. van de Geijn, *Using PLAPACK*, The MIT Press, 1997

Related Journal Papers

- Jack Dongarra, Robert van de Geijn and David Walker, "Scalability Issues Affecting the Design of a Dense Linear Algebra Library," Special Issue on Scalability of Parallel Algorithms, *Journal of Parallel and Distributed Computing*, Vol. 22, No. 3, Sept. 1994.
- Tom Cwik, Robert van de Geijn, and Jean Patterson, "The Application of Parallel Computation to Integral Equation Models of Electromagnetic Scattering," *Journal of the Optical Society of America A*, Vol. 11, No. 4, pp. 1538–1545, April 1994
- Jack Dongarra and Robert van de Geijn, "Reduction to Condensed Form on Distributed Memory Architectures," in *Parallel Computing*, 18, pp. 973–982, 1992.

- Po Geng, J. Tinsley Oden, and Robert van de Geijn, “Massively Parallel Computation for Acoustical Scattering Problems using Boundary Element Methods,” *Journal of Sound and Vibration,* 191 (1), pp. 145-165.
- Almadena Chtchelkanova, John Gunnels, Greg Morrow, James Overfelt, and Robert van de Geijn, “Parallel Implementation of BLAS: General Techniques for Level 3 BLAS,” *Concurrency: Practice and Experience* to appear.
- Brian Grayson and Robert van de Geijn, “A High Performance Parallel Strassen Implementation,” *Parallel Processing Letters*, Vol 6, No. 1 (1996) 3-12.
- Robert van de Geijn and Jerrell Watts, “SUMMA: Scalable Universal Matrix Multiplication Algorithm,” *Concurrency: Practice and Experience*, to appear.

Related Conference Publications

- Greg Baker, John Gunnels, Greg Morrow, Beatrice Riviere, and Robert van de Geijn, “PLAPACK: High Performance through High Level Abstraction,” Proceedings of ICPP98, to appear.
- John Gunnels, Calvin Lin, Greg Morrow, and Robert van de Geijn, ”A Flexible Class of Parallel Matrix Multiplication Algorithms”, Proceedings of IPPS98, 1998.
- Philip Alpatov, Greg Baker, Carter Edwards, John Gunnels, Greg Morrow, James Overfelt, Robert van de Geijn, Yuan-Jye J. Wu, ”PLAPACK: Parallel Linear Algebra Package – Design Overview” , Proceedings of SC97, 1997.
- Philip Alpatov, Greg Baker, Carter Edwards, John Gunnels, Greg Morrow, James Overfelt, Robert van de Geijn, ”PLAPACK: Parallel Linear Algebra Package,” in Proceedings of the SIAM Parallel Processing Conference, 1997.
- Wu, Y.-J. J., A. A. Alpatov, C. Bischof, and R. A. van de Geijn, ”A Parallel Implementation of Symmetric Band Reduction Using PLAPACK”, Proceedings of Scalable Parallel Library Conference, Mississippi State. (October 1996, 8 pages, PRISM WN#35)
- Ken Klimkowski and Robert van de Geijn, ”Anatomy of an out-of-core dense linear solver”, *Proceedings of the 1995 International Conference on Parallel Processing,* pp. 29–33.
- J. G. Lewis, D. G. Payne, and R. A. van de Geijn, “Matrix-Vector Multiplication and Conjugate Gradient Algorithms on Distributed Memory Computers,” *Scalable High Performance Computing Conference 1994*.
- J. G. Lewis and R. A. van de Geijn, “Implementing Matrix-Vector Multiplication and Conjugate Gradient Algorithms on Distributed Memory Multicomputers,” Supercomputing ’93.
- James Demmel, Jack Dongarra, Robert van de Geijn and David Walker, “LAPACK for Distributed Memory Architectures: The Next Generation,” in *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing,* Norfolk, March 1993.
- Jack Dongarra, Robert van de Geijn, and David Walker), “A Look at Scalable Dense Linear Algebra Libraries,” in *Proceedings of Scalable High Performance Concurrent Computing ’92 (SHPCC92)*, April 27-29, 1992.
- Anderson, E., Benzoni, A., Dongarra, J., Moulton, S., Ostrouchov, S., Tourancheau, B., and van de Geijn, R., “Basic Linear Algebra Communication Subprograms.” in *Sixth Distributed Memory Computing Conference Proceedings*, IEEE Computer Society Press, 1991, pp. 287–290.

- Anderson, E., Benzoni, A., Dongarra, J., Moulton, S., Ostrouchov, S., Tourancheau, B., and van de Geijn, R., “LAPACK for Distributed Memory Architectures: Progress Report.” in *the Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, Philadelphia, 1992, pp. 625–630.

8 Vita: Greg Morrow

Education

B.S. in Mathematics, Stanford University, June 1988. Ph.D. in Physics, University of Texas at Austin, May 1997.

Professional Experience

Postdoctoral Fellow, Department of Computer Sciences and Texas Institute for Computational and Applied Mathematics, University of Texas at Austin, May 1997–present.

Software Engineering Coop, Intel Scalable Systems Division, Beaverton, Oregon, 1995–1996.

Research Assistant, Departments of Physics and Computer Sciences, 1991–1995.

Related Publications

- Almadena Chtchelkanova, John Gunnels, Greg Morrow, James Overfelt, and Robert van de Geijn, "Parallel Implementation of BLAS: General Techniques for Level 3 BLAS," *Concurrency: Practice and Experience* to appear.
- Greg Baker, John Gunnels, Greg Morrow, Beatrice Riviere, and Robert van de Geijn, "PLAPACK: High Performance through High Level Abstraction," Proceedings of ICPP98, to appear.
- John Gunnels, Calvin Lin, Greg Morrow, and Robert van de Geijn, "A Flexible Class of Parallel Matrix Multiplication Algorithms", Proceedings of IPPS98, 1998.
- Philip Alpatov, Greg Baker, Carter Edwards, John Gunnels, Greg Morrow, James Overfelt, Robert van de Geijn, Yuan-Jye J. Wu, "PLAPACK: Parallel Linear Algebra Package – Design Overview" , Proceedings of SC97, 1997.
- Philip Alpatov, Greg Baker, Carter Edwards, John Gunnels, Greg Morrow, James Overfelt, Robert van de Geijn, "PLAPACK: Parallel Linear Algebra Package," in Proceedings of the SIAM Parallel Processing Conference, 1997.