D R A F T

Document for the Basic Linear Algebra Subprograms (BLAS) Standard

Basic Linear Algebra Subprograms Technical (BLAST) Forum

December 6, 1997

Contents

The discussion in each chapter is not limited only to the points listed, any related topics can be included at the discretion of the chapter authors. Final acceptance of each section will be decided at the next meeting using *Robert's Rules*. See BLAST Forum homepage for more detailed rules.

Chapter 4

Distributed-memory Dense BLAS Interface

4.1 Introduction to Distributed-memory Dense BLAS

4.1.1 Overview and Goals

There has been much interest in the past few years in developing versions of the BLAS for distributed-memory computers[47, 28, 3, 29, 14, 17, 15, 18, 7, 48]. Some of this research proposed parallelizing the BLAS[43, 47, 17, 15, 13, 18, 46, 48], and some implemented a few important BLAS routines[47, 41, 17, 15, 13, 18, 46, 48], such as matrix-matrix multiplication[31, 5, 36, 42, 4, 16, 49] or triangular system solve [34, 39, 40, 12]. This document has been inspired to a large extent by these research initiatives.

This document proposes to reuse the interface design of the dense part of the ScaLAPACK project [15, 13] as a basis for the distributed-memory dense BLAS interface. This choice is motivated by the following reasons.

- The simplicity of this interface allows for ease-of-use and similar calling sequence definitions in all targeted programming languages.
- This interface has been shown[13] to be effective for the developement of large and high-quality dense linear algebra software for distributed-memory computers.
- The broad functionality permitted by this interface enables, facilitates and encourages the development of current and related research projects such as the Distributed BLAS project[46], the Multicomputer Toolbox project[29, 7] and the PLAPACK project[48].
- The publically available implementation containing source code, testing and timing programs considerably facilitate the task of providing a reference implementation of the distributed-memory dense BLAS interface.
- A large number of machine and software vendors such as Fujitsu, Hewlett-Packard/Convex, Hitachi, IBM Parallel ESSL, NAG Numerical PVM and MPI library, NEC Scientific Software Library, SGI Cray Scientific Library, SUN Scientific Software Library and Visual Numerics (IMSL) have already adopted this interface design for their own products.

The distributed-memory dense BLAS standardization effort involved about 30 people from 20 organizations mainly from the United States and Europe. Most of the major vendors of concurrent computers were involved in this work, along with researchers from universities, government laboratories, and industry. The standardization process began with the BLAS Technical Workshop, sponsored by the University of Tennessee Knoxville, held November 13-14, 1995, in Knoxville, Tennessee. At this workshop the basic features essential to a standard distributed-memory dense BLAS interface were discussed, and a working group established to continue the standardization process.

The main advantages of establishing a distributed-memory dense BLAS standard are portability and ease-of-use. In a distributed-memory environment in which the higher level routines and/or abstractions are built upon lower level message-passing and computational routines the benefits of standardization are particularly apparent. Furthermore, the definition of distributed-memory dense basic linear algebra subprograms, such as those proposed here, provides vendors with a clearly defined base set of routines that they can implement efficiently, or in some cases provide hardware support for, thereby enhancing scalability.

The goal of the distributed-memory dense BLAS interface simply stated is to develop a widely used standard for writing message-passing programs performing dense basic linear algebra operations. As such the interface should establish a practical, portable, efficient and flexible standard for distributed-memory dense basic linear algebra operations.

A complete list of goals follows.

- Design an Application Programming Interface (API) (not necessarily for compilers or a system implementation library) well suited for distributed-memory dense basic linear algebra computations.
- Allow efficient communication and computation: minimizing communication startup overhead and volume, while maximizing load balance and local computational performance.
- Allow for re-use of existing message-passing interface standard[30] as well as local basic linear algebra computational kernels such as the *de facto* standard BLAS.
- Allow for implementations that can be used in a heterogeneous environment.
- Allow convenient Fortran 77, Fortran 90, High Performance Fortran (HPF), C and C++ bindings for the interface.
- Define an interface that is not too different from current practice and provide extensions that allow greater flexibility.
- Define an interface that can be implemented on many vendors' platforms, with no significant changes in the underlying system software.
- Semantics of the interface should be language- and data-distribution independent.
- The interface should be designed to allow for thread-safety.

4.1.2 Who Should Use This Standard ?

This standard is intended for use by all those who want to write portable programs performing dense linear algebra operations in Fortran 77, Fortran 90, High Performance Fortran (HPF), C or C++. This includes individual application programmers, developers of dense linear algebra software designed to run on parallel machines, and creators of computational environment and tools. In order to be attractive to this wide audience, the standard must provide a simple, easy-to-use interface for the basic user while not semantically precluding the high-performance computation and communication operations available on advanced machines.

4.1.3 What Platforms Are Targets for Implementation ?

The attractiveness of the distributed-memory dense BLAS at least partially stems from its wide portability as well as the common occurence of dense linear algebra operations in numerical simulations. These programs may run on distributed-memory multiprocessors, networks of workstations, and combinations of all of these. In addition, shared-memory implementations are possible. The message passing paradigm will not be made obsolete by architectures combining the shared- and distributed- memory views, or by increases in network speeds. It thus should be both possible and useful to implement this standard on a great variety of machines, including those "machines", parallel or not, connected by a communication network.

The distributed-memory dense BLAS interface provides many features intended to improve performance on scalable parallel computers with specialized interprocessor communication hardware. Thus, we expect that native, high-performance implementations of this interface will be provided on such machines. At the same time, implementations of this standard on top of MPI or PVM will provide portability to workstation clusters and heterogeneous networks of workstations.

4.1.4 What Is Included in the Standard ?

The standard includes:

- A set of basic dense linear algebra computational operations
- Data-redistribution operations
- Environmental management and inquiry
- Bindings for Fortran 77 and C that can also be used in respectively Fortran 90 and C++ programs
- Bindings for High Performance Fortran (HPF)

4.1.5 What Is Not (Yet ?) Included in the Standard ?

The standard does not (yet) specify:

- Debugging facilities
- Specific I/O and data generation operations
- Out-of-core computational routines
- Packed and banded storage computational routines
- Extended precision computational routines

There are many features and interface variants that have been considered and not included in this standard. This happened for a number of reasons, one of which is the time constraint that was self-imposed in finishing the standard. Features that are not included can always be offered as extensions by specific implementations. Perhaps future version of the distributed-memory dense BLAS interface will address some of these issues.

4.1.6 Organization of This Chapter

The following is a list of the remaining sections in this chapter, along with a brief description of each.

- Section **??**, Terminology and Conventions, explains notational terms and conventions used throughout this chapter.
- Section **??**, Functionality, describes the operations performed by the distributed-memory dense BLAS functions,
- Section **??**, Language and Data-distribution Independent Specifications, defines the operations of the distributed-memory dense BLAS independently from the data-decomposition scheme used for distributing the operands. Matrix-matrix multiply, matrix-vector multiply and dot-product are found there, along with many associated functions designed to make the basic linear algebra computational operation powerful, easy to use and efficient.
- Section **??**, Language- and Data-Distribution Dependent Specifications, defines the Fortran 77 and C bindings of the distributed-memory dense BLAS functions for the two-dimensional block-cyclic and block-cartesian data decomposition schemes.
- Section **??**, A Reference Implementation, describes a model implementation of the distributed-memory dense BLAS as well as testing and timing programs for the two-dimensional block-cyclic mapping that aims at supporting and encouraging the use of the distributed-memory dense BLAS.
- Section **??**, Language Bindings, gives specific syntax in Fortran 77 and C, for all distributed-memory dense BLAS functions for the two-dimensional block-cyclic and block-cartesian data-decomposition schemes.
- The distributed-memory dense BLAS Index is a simple index showing the location of the precise definition of each function defined by this chapter, together with both C and Fortran 77 bindings.

Bibliography

[1] *The EM Symmetric Sparse Supernodal Format.* Elegant Mathematics, Inc., Bothell, Washington, 1993.

[2] *The EM Unsymmetric Sparse Supernodal Format.* Elegant Mathematics, Inc., Bothell, Washington, 1993.

[3] M. Aboelaze, N. Chrisochoides, and E. Houstis. The Parallelization of Level 2 and 3 BLAS Operations on Distributed Memory Machines. Technical Report CSD-TR-91-007, Purdue University, West Lafayette, IN, 1991.

[4] R. Agarwal, S. Balle, F. Gustavson, M. Joshi, and P. Palkar. A Three-Dimensional Approach to Parallel Matrix Multiplication. *IBM Journal of Research and Development*, 39(5):575–582, 1995.

[5] R. Agarwal, F. Gustavson, and M. Zubair. A High Performance Matrix Multiplication Algorithm on a Distributed-Memory Parallel Computer, Using Overlapped Communication. *IBM Journal of Research and Development*, 38(6):673–681, 1994.

[6] E. Anderson, Z. Bai, C. H. Bischof, J. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorensen. *LAPACK Users' Guide.* SIAM, Philadelphia, PA, USA, 2nd edition, 1995. (Also available in Japanese, published by Maruzen, Tokyo, translated by Dr Oguni).

[7] P. Bangalore. The Data-Distribution-Independent Approach to Scalable Parallel Libraries. Master's thesis, Mississippi State University, 1995.

[8] R. E. Bank and D. J. Rose. Marching algorithms for elliptic boundary value problems. the constant coefficient case. *SIAM Journal of Numerical Analysis*, 14:792–829, 1977.

[9] Randolph E. Bank and Craig C. Douglas. Sparse matrix multiplication package (SMMP). *Advances in Computational Mathematics*, 1, 1993.

[10] B. Bialecki and G. Fairweather. Matrix decomposition algorithms in orthogonal spline collocation for separable elliptic boundary value problems. *SIAM J. Sci. Comput.*, 16:330–347, 1995.

[11] J. Bilmes, K. Asanovic, C. Chin, and J. Demmel. Optimizing matrix multiply using phipac: a portable, high-performance, ansi c coding methodology. In *Proceedings of the International Conference on Supercomputing.* ACM SIGARC, 1997. Vienna, Austria.

[12] R. Bisseling and J. van der Vorst. Parallel Triangular System Solving on a mesh network of Transputers. *SIAM Journal on Scientific and Statistical Computing*, 12:787–799, 1991.

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

[14] R. Brent and P. Strazdins. Implementation of BLAS Level 3 and LINPACK Benchmark on the AP1000. *Fujitsu Scientific and Technical Journal*, 5(1):61–70, 1993.

[15] J. Choi, J. Dongarra, S. Ostrouchov, A. Petitet, D. Walker, and R. C. Whaley. A Proposal for a Set of Parallel Basic Linear Algebra Subprograms. In J. Dongarra, K. Masden, and J. Waśniewski, editors, *Applied Parallel Computing*, pages 107–114. Springer Verlag, 1995. (also LAPACK Working Note No.100).

[16] J. Choi, J. Dongarra, and D. Walker. PUMMA: Parallel Universal Matrix Multiplication Algorithms on Distributed Memory Concurrent Computers. *Concurrency: Practice and Experience*, 6(7):543–570, 1994. (also LAPACK Working Note No.57).

[17] J. Choi, J. Dongarra, and D. Walker. PB-BLAS: A Set of Parallel Block Basic Linear Algebra Subroutines. *Concurrency: Practice and Experience*, 8(7):517–535, 1996.

[18] A. Chtchelkanova, J. Gunnels, G. Morrow, J. Overfelt, and R. van de Geijn. Parallel Implementation of BLAS: General Techniques for Level 3 BLAS. Technical Report TR95-49, Department of Computer Sciences, UT-Austin, 1995. Submitted to Concurrency: Practice and Experience.

[19] J. Demmel, J. R. Gilbert, and X. S. Li. An asynchronous parallel supernodal algorithm for sparse Gaussian elimination, February 1997. To appear in SIAM J. Matrix Anal. Appl., special issue on Sparse and Structured Matrix Computations and Their Applications (Also LAPACK Working Note 124).

[20] J. Demmel and X. Li. Faster numerical algorithms via exception handling. *IEEE Trans. Comput.*, 43:983–992, 1994.

[21] D. S. Dodson. Corrigendum: Remark on "Algorithm 539: Basic Linear Algebra Subroutines for FORTRAN usage". *ACM Trans. Math. Software*, 9:140, 1983. (See also [38] and [22]).

[22] D. S. Dodson and R. G. Grimes. Remark on algorithm 539: Basic Linear Algebra Subprograms for Fortran usage. *ACM Trans. Math. Software*, 8:403–404, 1982. (See also [38] and [21]).

[23] D. S. Dodson, R. G. Grimes, and J. G. Lewis. Sparse extensions to the FORTRAN Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 17:253–272, 1991. (Algorithm 692).

[24] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 16:1–28, 1990. (Algorithm 679).

[25] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. An extended set of FORTRAN Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 14:1–32, 399, 1988. (Algorithm 656).

[26] I. S. Duff, A. M. Erisman, and J. K. Reid. Direct methods for sparse matrices. In *Monographs on Numerical Analysis*. Oxford University Press, New York, 1986.

[27] Iain Duff, Michele Marrone, and Giuseppe Radicati. A proposal for user level sparse BLAS. Technical Report TR/PA/92/85, CERFACS, 1992.

[28] A. Elster. Basic Matrix Subprograms for Distributed Memory Systems. In D. Walker and Q. Stout, editors, *Proceedings of the Fifth Distributed Memory Computing Conference*, pages 311–316. IEEE Press, 1990.

[29] R. Falgout, A. Skjellum, S. Smith, and C. Still. The Multicomputer Toolbox Approach to Concurrent BLAS and LACS. In *Proceedings of the Scalable High Performance Computing Conference SHPCC-92*. IEEE Computer Society Press, 1992.

[30] Message Passing Interface Forum. MPI: A Message Passing Interface Standard. *International Journal of Supercomputer Applications and High Performance Computing*, 8(3–4), 1994.

[31] G. Fox, S. Otto, and A. Hey. Matrix Algorithms on a Hypercube I: Matrix Multiplication. *Parallel Computing*, 3:17–31, 1987.

[32] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, 28:505–535, 1974.

[33] F. G. Gustavson. Two fast algorithms for sparse matrices: Multiplication and permuted transposition. *ACM Transactions on Mathematical Software*, 4(3):250–269, 1978.

[34] M. Heath and C. Romine. Parallel Solution Triangular Systems on Distributed Memory Multiprocessors. *SIAM Journal on Scientific and Statistical Computing*, 9:558–588, 1988.

[35] M. A. Heroux. A proposal for a sparse BLAS toolkit. Technical Report TR/PA/92/90, CERFACS, 1992.

[36] S. Huss-Lederman, E. Jacobson, A. Tsao, and G. Zhang. Matrix Multiplication on the Intel Touchstone DELTA. *Concurrency: Practice and Experience*, 6(7):571–594, 1994.

[37] L. Kaufman and D. D. Warner. High-order, fast direct methods for separable elliptical equations. *SIAM Journal of Numerical Analysis*, 21:672–695, 1984.

[38] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh. Basic Linear Algebra Subprograms for FORTRAN usage. *ACM Trans. Math. Software*, 5:308–323, 1979. (Algorithm 539. See also [22] and [21]).

[39] G. Li and T. Coleman. A Parallel Triangular Solver for a Distributed-Memory Multiprocessor. *SIAM Journal on Scientific and Statistical Computing*, 9(3):485–502, 1988.

[40] G. Li and T. Coleman. A New Method for Solving Triangular Systems on Distributed-Memory Message-Passing Multiprocessor. *SIAM Journal on Scientific and Statistical Computing*, 10(2):382–396, 1989.

[41] W. Lichtenstein and S. L. Johnsson. Block-Cyclic Dense Linear Algebra. *SIAM Journal on Scientific and Statistical Computing*, 14(6):1259–1288, 1993.

[42] K. Mathur and S. L. Johnsson. Multiplication of Matrices of Arbitrary Shapes on a Data Parallel Computer. *Parallel Computing*, 20:919–951, 1994.

[43] O. McBryan and E. van de Velde. Matrix and Vector Operations on Hypercube Parallel Processors. *Parallel Computing*, 5:117–126, 1987.

[44] R. Pozo. Template numerical toolkit (tnt), 1995. `http://math.nist.gov/tnt`.

[45] Y. Saad. SPARSKIT: a basic tool kit for sparse matrix computations. Preliminary Version.

[46] P. Strazdins. A High Performance, Portable Distributed BLAS Implementation. In *Proceedings of the Sixth Parallel Computing Workshop*, Fujitsu Parallel Computing Center, 1996.

[47] Thinking Machines Corporation. *CMSSL for Fortran*, 1990.

[48] R. van de Geijn. *Using PLAPACK: Parallel Linear Algebra Package.* The MIT Press, Cambridge, Massachusetts, 1997.

[49] R. van de Geijn and J. Watts. SUMMA: Scalable Universal Matrix Multiplication Algorithm. Technical Report UT CS-95-286, LAPACK Working Note No.96, University of Tennessee, 1995.

[50] T. et al. Yang. Distributed memory parallel sparse lu. Ucsb preprint, University of California, Santa Barbara.