Summary of SuperLU

Xiaoye S. Li[1]

1 Overview of the Package

SuperLU is a general-purpose library for the direct solution of large, sparse systems of linear equations on high performance machines. It is especially targeted for non-symmetric, non-definite systems. The library routines perform an LU factorization with numerical pivoting and triangular solutions through forward and back substitutions. The factorization can be applied to non-square matrices. In addition, the driver routines contain the functionalities of equilibrating the matrix, reordering the rows and columns of the matrix for stability and sparsity, iterative refinement, estimating the condition number, and computing the forward and componentwise backward error bounds. The solver supports both real and complex data types.

SuperLU is implemented in ANSI C, with examples of calling it from Fortran. The complete package consists of three parts: `SuperLU` for sequential machines, `SuperLU_MT` for shared-memory machines, and `SuperLU_DIST` for distributed-memory machines. The standard Pthreads is used in `SuperLU_MT` and MPI is used in the `SuperLU_DIST`. The users can use them separately or simultaneously in a single application. A comprehensive overview of all three packages can be found in [8].

2 Main Features of the Algorithms and Implementation

2.1 `SuperLU` for sequential machines.

The main references for this work are [2, 6]. The algorithms and implementation techniques are summarized below.

- Left-looking factorization; partial pivoting, and possibly with diagonal preference to better preserve sparsity.
- Identify unsymmetric supernodes to perform most of the numerical computation using dense matrix kernels.
- Factorize several consecutive columns (a panel) at a time to improve locality (effectively using BLAS-3).
- Use 2D partitioning of supernodes (loop reordering) to avoid cache thrashing; identify blocking parameters that are tunable by users for different cache size.
- Use Gilbert-Peierls' depth-first search and Eisenstat-Liu's symmetric pruning, in the context of panel update, to speed up symbolic factorization.

2.2 `SuperLU_MT`

The main references for this work are [3, 6]. The code has been tested on Sun, DEC Alpha, SGI Power Challenge, SGI Origin, Convex Exemplar, and Cray C90/J90. For each of these machines, a `Makefile` is provided to compile the code using machine-specific pragmas for parallelization. Alternatively, the code can be compiled using POSIX threads (Pthreads), which is a widely portable interface. A `Makefile` is also provided for this interface. All the implementation techniques used in `SuperLU` are also used in `SuperLU_MT`. The additional parallelization techniques are summarized below.

- Use an asychronous and barrier-free scheduling algorithm to schedule two types of parallel tasks to achieve a high level of concurrency. One such task is factorizing the independent

[1]Lawrence Berkeley National Laboratory, MS 50F1650, One Cyclotron Road, Berkeley, CA 94720. xsli@lbl.gov.

panels in the disjoint subtrees of the column elimination tree. Another task is updating a panel by previously computed supernodes. The scheduler facilitates the smooth transition between the two types of tasks, and maintains load balance dynamically.

- In symbolic factorization, a non-blocking algorithm is used to perform depth-first search and symmetric pruning in parallel.

2.3 SuperLU_DIST

The main references for this work are [4, 5, 7]. The code has been tested on many platforms, such as Cray T3E, IBM SP, and Linux clusters. Most of the implementation techniques used in SuperLU are also used in SuperLU_DIST. The main differences are in pivoting strategy and matrix distribution, which are designed to achieve high scalability, and are summarized below.

- Right-looking factorization; use elimination DAGs to identify task and data dependencies.
- Before factorization, pre-permute the rows of the matrix so that the diagonal has entries of large magnitude, using a weighted bipartite matching algorithm; during factorization, allow half-precision perturbation to the small diagonal entry.
- Use 2D (irregular) block-cyclic mapping based on supernodal structure,
- Parallel pipelined factorization algorithm to better overlap communication with computation.

3 Large Scientific Applications Using SuperLU_DIST

We describe two scientific applications in which SuperLU_DIST has played a critical role. The first application is in the solution of a long-standing problem of scattering in a quantum system of three charged particles. This requires solving the complex, nonsymmetric, and very ill-conditioned linear systems. The largest system solved is of order 8 million. SuperLU_DIST is used in building the block diagonal preconditioners for the CGS iterative solver. The number of CGS iterations ranges between 12 to 35. Since each CGS iteration requires two preconditioning steps, 24 to 70 triangular solutions for the diagonal blocks are required. For a block of size 1 million, SuperLU_DIST takes 1209 seconds to factorize using 64 processors of the IBM SP at NERSC, and it takes 26 seconds to perform triangular solution. The total execution time is about 1 hour. See [1] for more details. The scientific breakthrough result was reported in a cover article of *Science* [9].

More recently, we have been collaborating with researchers at the Stanford Linear Accelerator Center to develop alternative eigensolvers for Omega3P, a widely used electromagnetics code in accelerator design. In this application the interior eigenvalues and eigenvectors of a large sparse generalized eigenvalue problem are needed. We integrated SuperLU_DIST with PARPACK to construct a shift-and-invert eigensolver. For a system of order 1.3 million, PARPACK needs about 4.5 solutions for each eigenpair. For each solution, SuperLU_DIST takes 39 seconds using 32 processors of the IBM SP at NERSC. The factorization is done once, and takes 553 seconds. The total time for finding 10 interior eigenpairs is 42 minutes.

4 Other Remarks

The SuperLU package has been widely adopted in research and commercial use. It has several thousands users worldwide. Most users use SuperLU or SuperLU_DIST. Several popular vendors, including Boeing BCSLIB-EXT, HP Math Library, NAG, and SciPython, have incorporated SuperLU into their mathematical libraries, The widespread use of the software is a proof that the

software represents an important resource for the computational science community. Moreover, it shows that the software is easy to use.

The software will be under continuous support and maintenance by the authors. The source code and documentation are available at `http://crd.lbl.gov/~xiaoye/SuperLU/`.

References

[1] M. Baertschy and X. S. Li. Solution of a three-body poblem in quantum mechanics. In *Proceedings of SC2001*, Denver, Colorado, November 10–16 2001.

[2] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[3] James W. Demmel, John R. Gilbert, and Xiaoye S. Li. An asynchronous parallel supernodal algorithm for sparse gaussian elimination. *SIAM J. Matrix Analysis and Applications*, 20(4):915–952, 1999.

[4] Iain S. Duff and Jacko Koster. On algorithms for permuting large entries to the diagonal of a sparse matrix. *SIAM J. Matrix Analysis and Applications*, 22(4):973–996, 2001.

[5] Jacko Koster. On the parallel solution and the reordering of unsymmetric sparse linear systems. *Ph.D. Thesis*, CERFACS, France, November, 1997.

[6] Xiaoye S. Li. Sparse Gaussian elimination on high performance computers *Ph.D. Thesis*, U.C. Berkeley, September, 1996.

[7] Xiaoye S. Li and James W. Demmel, SuperLU_DIST: A Scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Trans. Mathematical Software*, 29 (2), June 2003, pp. 110–140.

[8] Xiaoye S. Li, An Overview of SuperLU: Algorithms, Implementation, and User Interface, *ACM Trans. Mathematical Software*, 31 (3), September 2005, pp. 302–325.

[9] T. N. Rescigno, M. Baertschy, W. A. Isaacs, and C. W. McCurdy. Collisional breakup in a quantum system of three charged particles. *Science*, 286:2474–2479, December 24, 1999.