# SuperLU – High-performance Library to Solve Sparse Linear Systems

Xiaoye S. Li
xiaoye@nersc.gov
LBNL/NERSC

ACTS Toolkit Workshop
September 28, 2000



## Outline

- Essential components of the sparse direct solvers
- The role of supernode in SuperLU
- Sequential and shared memory algorithms
- A Scalable algorithm for MPPs
- Summary

## What is SuperLU ?

- Solve large, sparse linear system

$$Ax = b$$

(Example: $A$ of size $10^5$-by-$10^5$, only $10 \sim 100$ nonzeros per row.)

- Algorithm: Gaussian elimination (LU factorization, $A = LU$), followed by lower/upper triangular solutions.
  - Store only nonzeros and perform operations only on the nonzeros.

- Efficient implementation for modern high-performance computers; Software highly portable on many platforms.



## Software Status

| | SuperLU | SuperLU_MT | SuperLU_DIST |
|---|---|---|---|
| Platform | Uniprocessor | SMP | Message-passing |
| Language | C (all callable from Fortran) | C + POSIX thread | C + MPI |
| Data type | real/complex, single/double | real, double | real/complex, double |

- Source code, Users' Guide, and papers available: http://www.nersc.gov/∼xiaoye/SuperLU

## Direct Solvers for Sparse Linear Systems

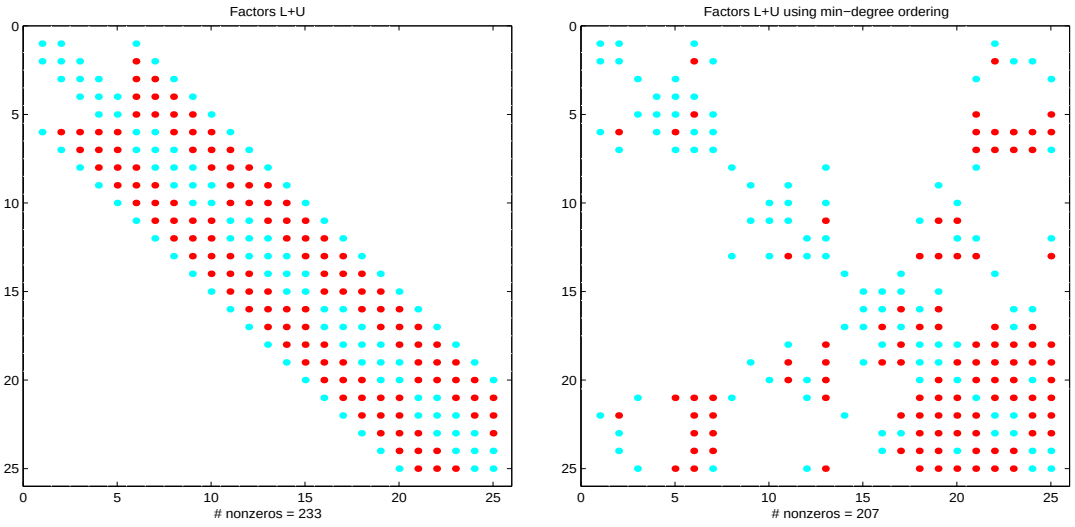
- Gaussian elimination ($LU$, $LL^T$, $LDL^T$ factorizations), followed by lower/upper triangular solutions.
  - Dense: $PA = LU$ permutation for stability
  - Sparse: $P_r A P_c^T = LU$ permutations for stability and sparsity of $L$, $U$

- Distinct steps for sparse matrices.
  1. Order equations & variables to minimize fill in the $L$, $U$ factors
     heuristics based on combinatorics
  2. Symbolic factorization
     set up data structures and allocate memory for $L$, $U$
  3. Numerical factorization: usually dominates total time
     how to pivot?
  4. Triangular solves: usually $< 5\%$ time



## Sparse GE : Fill-in

- Original zero entry $a_{ij}$ becomes nonzero in L or U



## How to Pivot?

- Pivoting traditionally used to control element growth in L & U for numerical stability.
- Example: partial pivoting: $PA = LU$ (GEPP).
  - used in sequential SuperLU and SuperLU_MT
- Partial pivoting implies
  - dynamic change of fill patterns of L & U
    $\Longrightarrow$ interleave symbolic & numerical factorizations
  - losts of communication with small messages $\Longrightarrow$ slow on parallel machines with high latency
- Static pivoting used in SuperLU_DIST (GESP).
  - pivot before numerical factorization so data structures static
  - accommodate possible pivot growth during factorization without changing data structures

  $\Longrightarrow$ symbolic & numerical factorizations decoupled



## Ordering for Sparse Cholesky

- Local greedy heuristics.
  - Minimum degree [Tinney/Walker '67, George/Liu '79, Liu '85, Amestoy/Davis/Duff '94, Ashcraft '95, Duff/Reid '95]
  - Minimum deficiency (fill-in) [Tinney/Walker '67, Ng/Raghavan '97]
- Global graph partitioning heuristics.
  - Nested dissection [George '73]
  - Multilevel schemes [Hendrickson/Leland '94, Karypis/Kumar '95]
  - Spectral bisection [Simon *et al.* '90-'95]
  - Geometric and spectral bisection [Chan/Gilbert/Teng '94]
- Hybrid of the above two [Ashcraft/Liu '96, Hendrickson/Rothberg'97].

## Ordering for Unsymmetric LU with Partial Pivoting

- Symmetric ordering for Cholesky of $A^T A$.
  - If $R^T R = A^T A$ and $PA = LU$, then for any $P$, $struct(L+U) \subseteq struct(R^T + R)$. [George/Ng '87]
  - Making $R$ sparse tends to make $L+U$ sparse.
  - Strategy:
    1. Find a good symmetric ordering $P_c$ from $A^T A$
    2. Apply $P_c$ to columns of $A$

$$(AP_c^T)^T(AP_c^T) = P_c(A^T A)P_c^T$$

- Column minimum degree based solely on $A$.
  [Matlab, Larimore/Davis/Gilbert/Ng '98]

- Markowitz – unsymmetric variant of minimum degree.
  [Duff/Erisman/Reid '86 book]
  usually performed together with numeric factorization



## Ordering for Unsymmetric LU with Diagonal Pivoting

- Symmetric ordering for Cholesky of $A^T + A$.
  - If $R^T R = A^T + A$ and $A = LU$, then $struct(L+U) \subseteq struct(R^T + R)$.
  - Making $R$ sparse tends to make $L+U$ sparse.
  - Strategy:
    1. Find a good symmetric ordering $P_c$ from $A^T + A$
    2. Apply $P_c$ to both rows and columns of $A$: $P_c A P_c^T$

$$Struct(P_c A P_c^T) \subseteq Struct(P_c(A^T + A)P_c^T)$$

## Ordering Interface in SuperLU

- SuperLU distribution contains routines:
  - Form $A^T A$
  - Form $A^T + A$
  - MMD (Multiple Minimum Degree, courtesy of Joseph Liu)
  - COLAMD : http://www.netlib.org/linalg/colamd/

- You may use any other – Just input a permutation vector to SuperLU
  For example:
  - (Par)METIS : http://www-users.cs.umn.edu/~karypis/metis/
  - ...



## Symbolic Factorization

- Cholesky [George/Liu '81 book]
  - Use elimination graph of $L$ and its transitive reduction (elimination tree)
  - Complexity linear in output: $O(nnz(L))$

- $LU$
  - Use elimination graphs of $L$, $U$ and their transitive reductions (elimination DAGs) [Tarjan/Rose '78, Gilbert/Liu '93, Gilbert '94]
  - Improved by symmetric structure pruning [Eisenstat/Liu '92]
  - Improved by supernodes
  - Complexity greater than $nnz(L+U)$, yet much smaller than $flops(LU)$

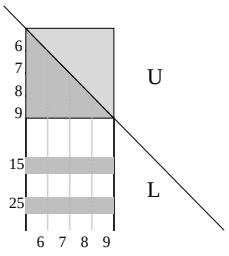
## Numerical Factorization

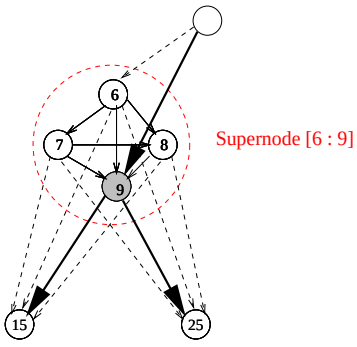
- Usually the most expensive step
- Recent improvements for
  - Superscalar processor and hierarchical memory system
  - Multiple processors



## Unsymmetric Supernode [Eisenstat/Gilbert/Liu '93]

- Exploit dense submatrices in the $L$ & $U$ factors of $PA = LU$




- Why are supernodes good?
  - Permit use of Level 3 BLAS
  - Reduce inefficient indirect addressing (scatter/gather)
  - Reduce symbolic time by traversing a coarser graph

## Supernode-Panel factorization

**for** column $j = 1$ **to** $n$ **step** $w$ **do**

$F(:, j{:}\, j+w-1) = A(:, j{:}\, j+w-1)$;

(1) Symbolic factorization [Gilbert/Peierls '88, Gilbert/Li '94]

- Determine which supernodes update $F(:, j{:}\, j+w-1)$

(2) Numeric update

**for** each updating supernode $(r{:}\, s) < j$ in order **do**

- Triangular solve

$U(r:s, j:j+w-1) =$

$L(r:s, r:s) \backslash F(r:s, j:j+w-1)$;

- Matrix update

$F(s+1:n, j:j+w-1) -=$

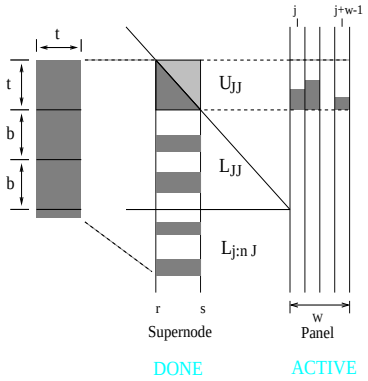
$L(s+1:n, r:s) * U(r:s, j:j+w-1)$;

**end for**;

(3) Inner factorization for $F(j:n, j+w-1)$

- Row pivoting for each column;
- Detect supernode boundary;
- Symmetric structure pruning; [Eisenstat/Liu '92]

**end for**;

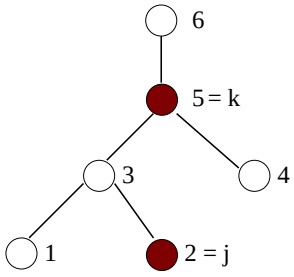





## Parallelism: Column Elimination Tree [Gilbert/Ng '93]

- Each column of the matrix has one vertex in the tree.
- Exhibits column dependencies during the elimination.
  1. If column $j$ updates column $k$, then $j$ is a descendant of $k$;
  2. Conversely, if $j$ is a descendant of $k$, column $j$ **may** or **may not** update column $k$ (depending on numerical pivoting).
- More accurate update edges are detected on the fly.
- Computing elimination tree takes time almost linear in $nnz(A)$.

$$A = \begin{pmatrix} 1 & \bullet & \bullet & \bullet \\ & 2 & \bullet & \bullet \\ & \bullet & 3 & \\ \bullet & & & 4 & \bullet \\ & & & & 5 & \bullet \\ \bullet & & & \bullet & & 6 \end{pmatrix}$$



## The Shared Memory Scheduling Loop [Demmel/Gilbert/Li '97]

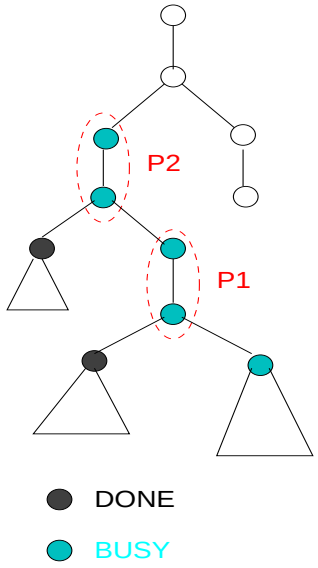
Shared task queue initialized with leaves;

while ( there are more panels ) do
  panel := GetTask ( queue );

  (1) panel_symbolic_factor( panel );
    • skip all BUSY descendant supernodes;

  (2) panel_numeric_factor( panel );
    • updates from all DONE supernodes;
    • wait for BUSY supernodes to become DONE;

  (3) inner factorization( panel );
end while;

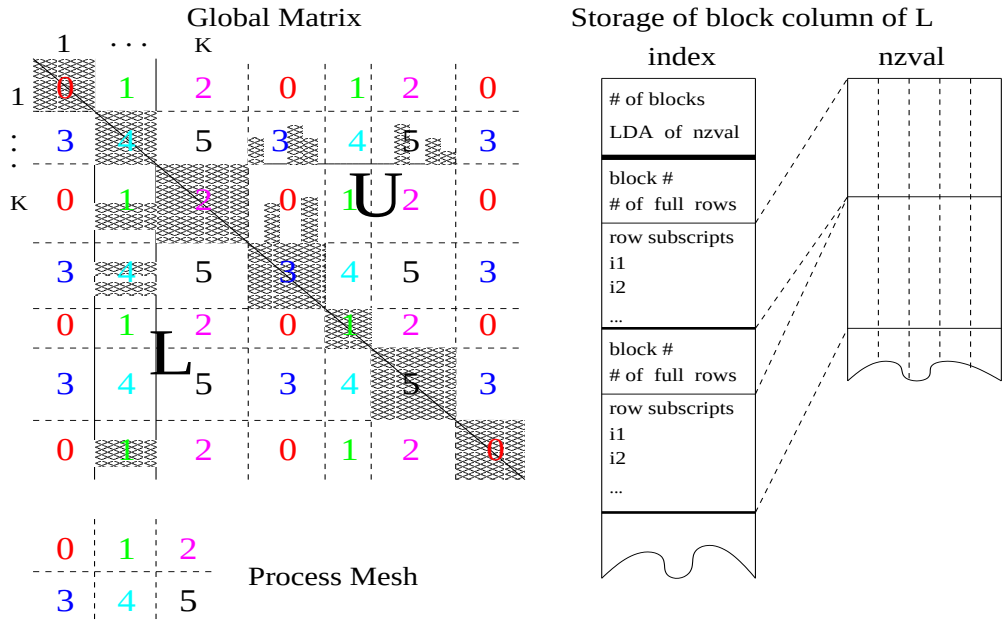





## SuperLU_DIST – GESP Algorithm

1. Row/column equilibration: $A \leftarrow D_r \cdot A \cdot D_c$
   • the largest entry of each row/column is 1
2. Permute rows to maximize diagonal: $A \leftarrow P_r \cdot A$
   • weighted bipartite matching algorithm [Duff/Koster '98]
3. Permute rows/columns to maximize sparsity & parallelism: $A \leftarrow P_c \cdot A \cdot P_c^T$
   • minimum degree, nested dissection ... on $A^T + A$
4. Factorize $A = LU$ in parallel: 2D irregular block cyclic layout
   • increase tiny pivots $a_{ii}$ to $\sqrt{\varepsilon}||A||$
5. Triangular solves in parallel
   • iterative refinement if needed

# Distributed Data Structures: 2D Block Cyclic Layout





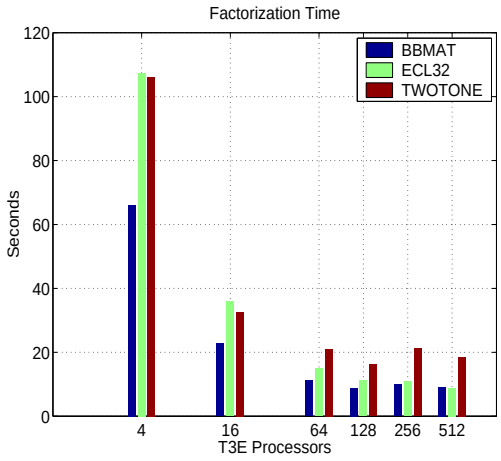

# High Performance

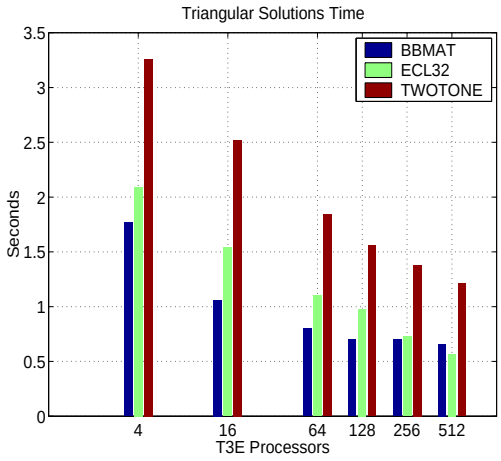
- *Sequential SuperLU*
  - Enhance data reuse in memory hierarchy by working with dense submatrices in the sparse factored matrices.
  - Achieved up to 40% of the theoretical Megaflop rate on workstations.
- *SuperLU_MT*
  - Exploit both coarse and fine grain parallelism; Employ dynamic scheduling to minimize parallel runtime.
  - Achieved up to 10 fold speedup on medium-size SMPs.
- *SuperLU_DIST*
  - Enhance scalability by the new static pivoting and matrix distribution methods.
  - Achieved up to 100 fold speedup, and 11 Gigaflop rate on 512-PE T3E.

# Example Matrices

Minimum degree ordering on $A^T + A$

| | Discipline | Symm | Order | $nnz(A)$ | $nnz(L+U)$ $(10^6)$ | Flops $(10^9)$ |
|---|---|---|---|---|---|---|
| BBMAT | fluid flow | 0.54 | 38,744 | 1,771,722 | 40.2 | 31.2 |
| ECL32 | device sim. | 0.93 | 51,993 | 380,415 | 42.7 | 68.4 |
| TWOTONE | circuit sim. | 0.43 | 120,750 | 1,224,224 | 11.9 | 8.0 |



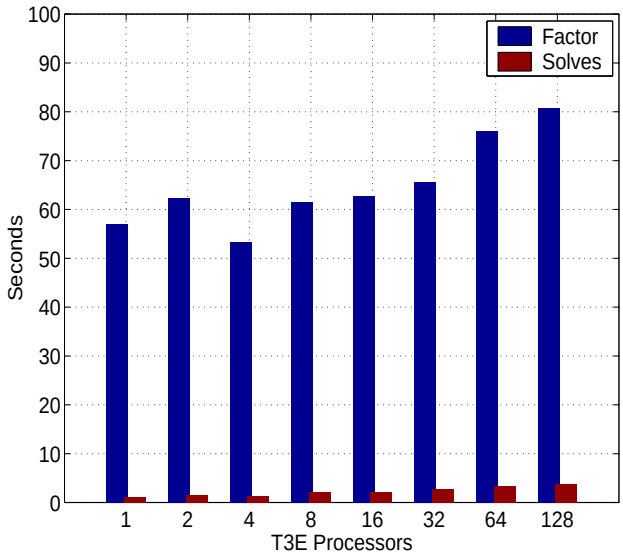






# Scalability on T3E

3D cubic grids, 11 points stencil.

Grid size increases with number of processors, such that flops per processor roughly constant: 29, 33, 36, 41, 46, 51, 57, 64.



## Quantum Chemistry Application Using SuperLU

- Study quantum scattering of 3 charged particles.
  (Recigno, Baertschy, Issacs & McCurdy, LBNL/LLNL)
- Complex unsymmetric linear systems, largest of order 5 million.
- SuperLU is used to build block Jacobi preconditioner in a CGS solver.
- Preconditioners
  - of order 209,764, in 2 minutes, on 16 PE Cray T3E at NERSC
  - of order 1,792,921, in 48 minutes, on 24 PE ASCI Blue-Pacific at LLNL
- These represent the first "exact" solution to a quantum mechanical 3-body Coulomb problem.
- Their historic result was featured on the cover of *Science*, Dec. 24, 1999.



## Summary – Content of the SuperLU Package

- LAPACK-style interface
  - Simple and expert driver routines
  - Computational routines
  - Comprehensive testing routines
- Functionalities
  - Minimum degree ordering [MMD, Liu '85] applied to $A^TA$ or $A + A^T$
  - User-input to control pivoting
    - pre-assigned row and/or column permutations
    - partial pivoting with threshold
  - Solving transposed system
  - Equilibration
  - Condition number estimation
  - Iterative refinement to improve accuracy
  - Componentwise error bounds [Skeel '79, Arioli/Demmel/Duff '89]