



---

# **SuperLU : High-performance Library to Solve Sparse Linear Systems**

X. Sherry Li  
[xiaoye@nersc.gov](mailto:xiaoye@nersc.gov)  
LBNL-NERSC

WORKSHOP ON THE ACTS TOOLKIT:  
High Performance in Numerical Libraries for Science and Engineering  
Los Alamos Computer Science Institute Symposium  
October 15-18, 2001



---

## 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 high-performance cache-based and parallel computers;  
Software highly portable on many platforms.



---

## Software Status

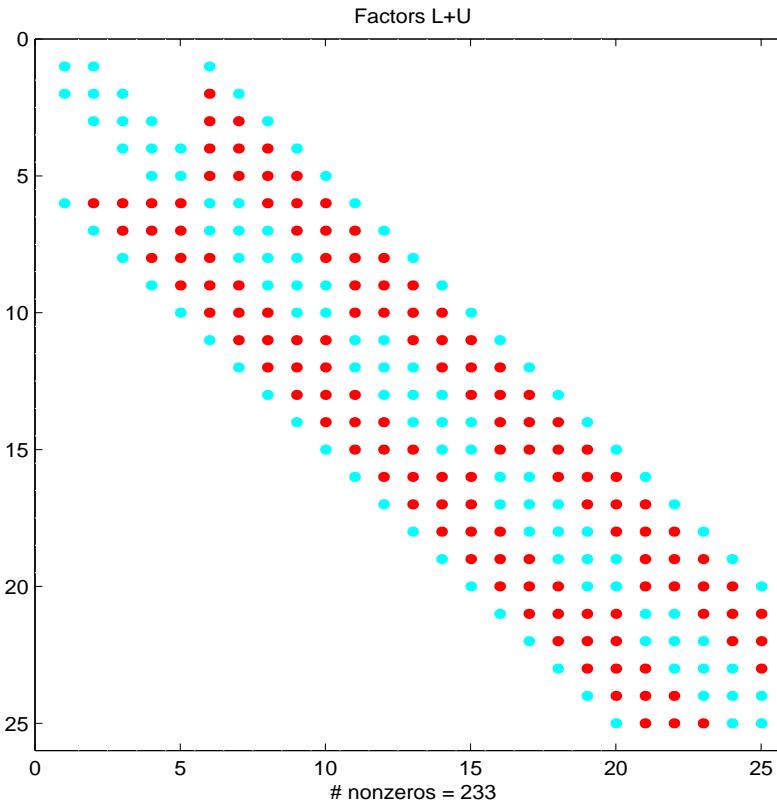
	SuperLU	SuperLU_MT	SuperLU_DIST
Platform	Uniprocessor	SMP	Message-passing
Language	C  (all callable from Fortran)	C + POSIX thread	C + MPI
Data type	real/complex, single/double	real, double	real/complex, double

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

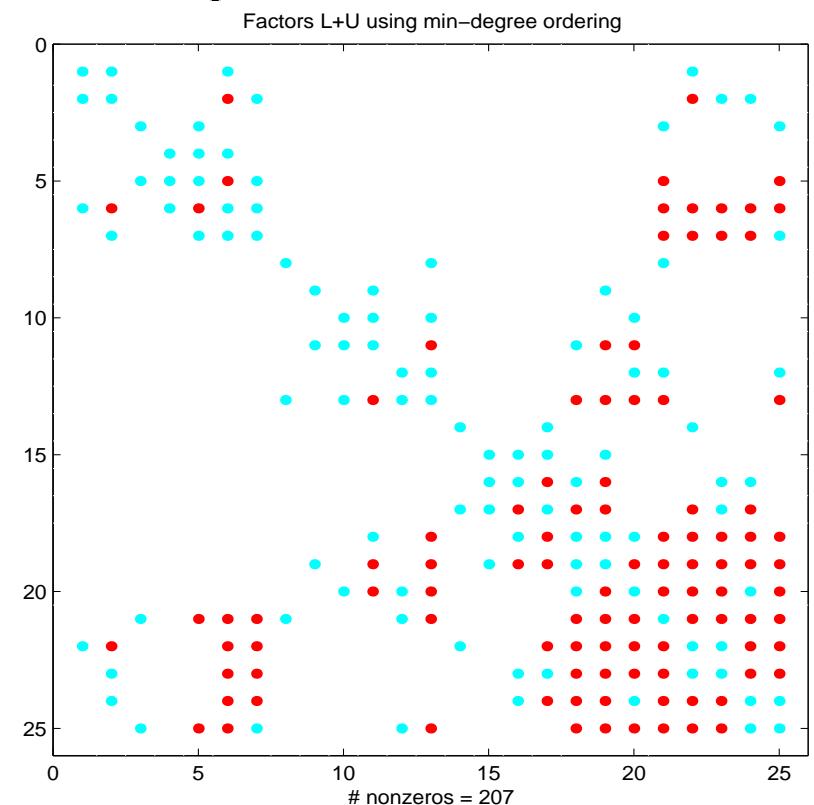
## Sparse GE : Fill-in

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

Natural Order: nonzeros = 233



Min. Degree Order: nonzeros = 207





---

## 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



---

## How to Pivot?

- Goal of pivoting: 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
    - ⇒ interleave symbolic & numerical factorizations
  - lots of communication with small messages
    - ⇒ 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
    - ⇒ 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 row permutation  $P$ ,  $\text{struct}(L + U) \subseteq \text{struct}(R^T + R)$ . [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  $A$  :  $\bar{A} = AP_c^T$

$$\bar{A}^T \bar{A} = (AP_c^T)^T (AP_c^T) = P_c(A^T A)P_c^T$$

- 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  $\text{struct}(L + U) \subseteq \text{struct}(R^T + R)$ .
  - 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  $A$  :  $\bar{A} = P_c A P_c^T$

$$\text{Struct}(\bar{A}) = \text{Struct}(P_c A P_c^T) \subseteq \text{Struct}(P_c (A^T + A) P_c^T)$$



## Ordering Comparison

		GEPP, COLAMD		GESP, AMD( $A^T + A$ )	
Matrix	Order	Fill ( $10^6$ )	Time	Fill ( $10^6$ )	Time
AF23560	23560	12.1	30.7	11.7	23.6
BRAMLEY1	17933	11.9	34.0	5.5	8.0
EX11	16614	18.9	82.5	11.4	22.0
FIDAPM11	22294	26.9	166.3	26.5	112.6
GOODWIN	7320	5.6	15.9	1.1	3.8
MEMPLUS	17758	4.4	36.4	0.14	1.4
ONETONE1	36057	4.8	22.2	3.1	19.1
WANG4	26068	26.2	197.2	10.5	31.4



---

## 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/>
  - Chaco
  - ...



---

## 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  $flops(LU)$



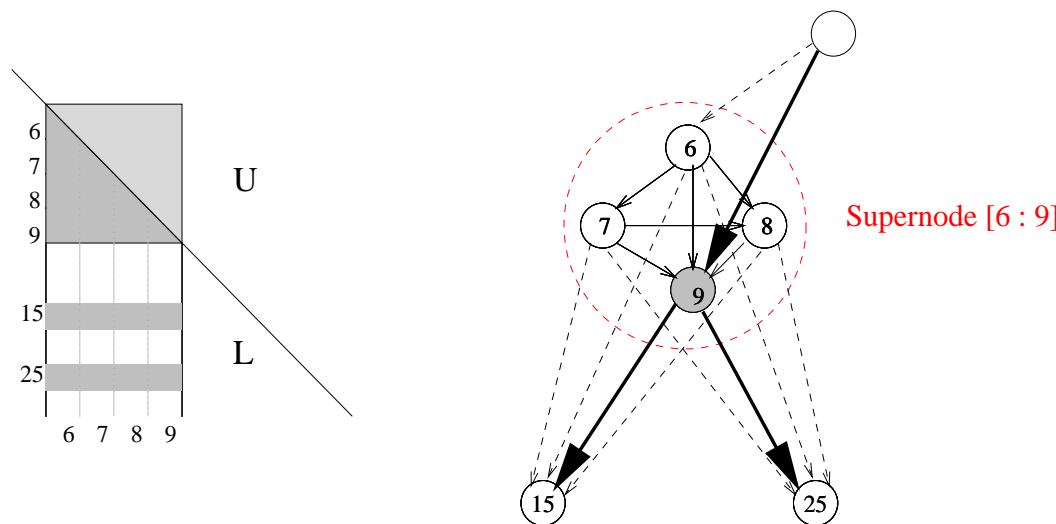
---

## 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 (Left-looking)

**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

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

- Matrix update

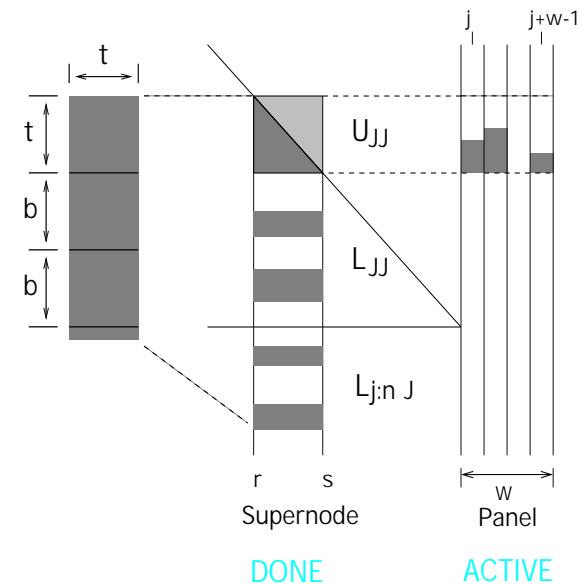
$$F(s+1:n, j:j+w-1) -= L(s+1:n, r:s) * U(r:s, j:j+w-1);$$

**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]

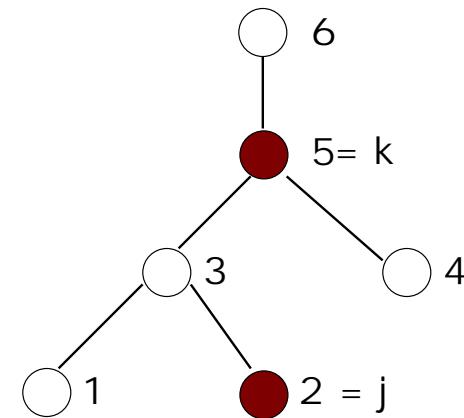
**end for;**



## 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  $nnz(A)$ .

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



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

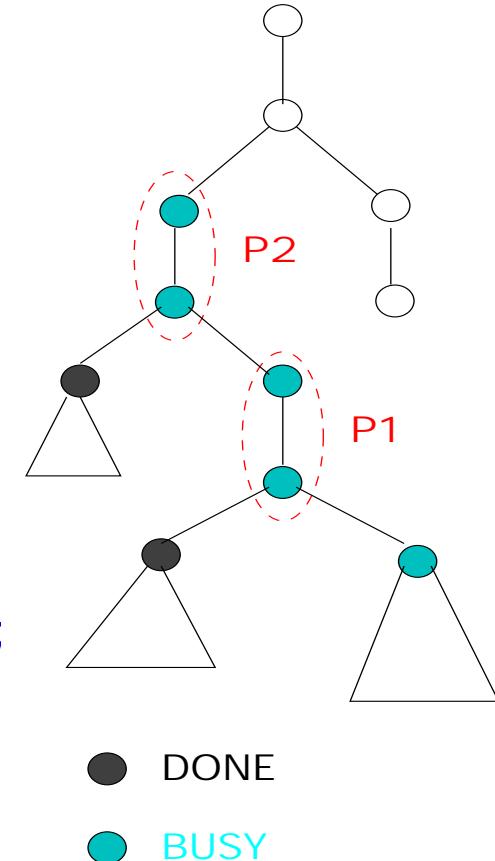
Shared task queue initialized with leaves;

```
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;



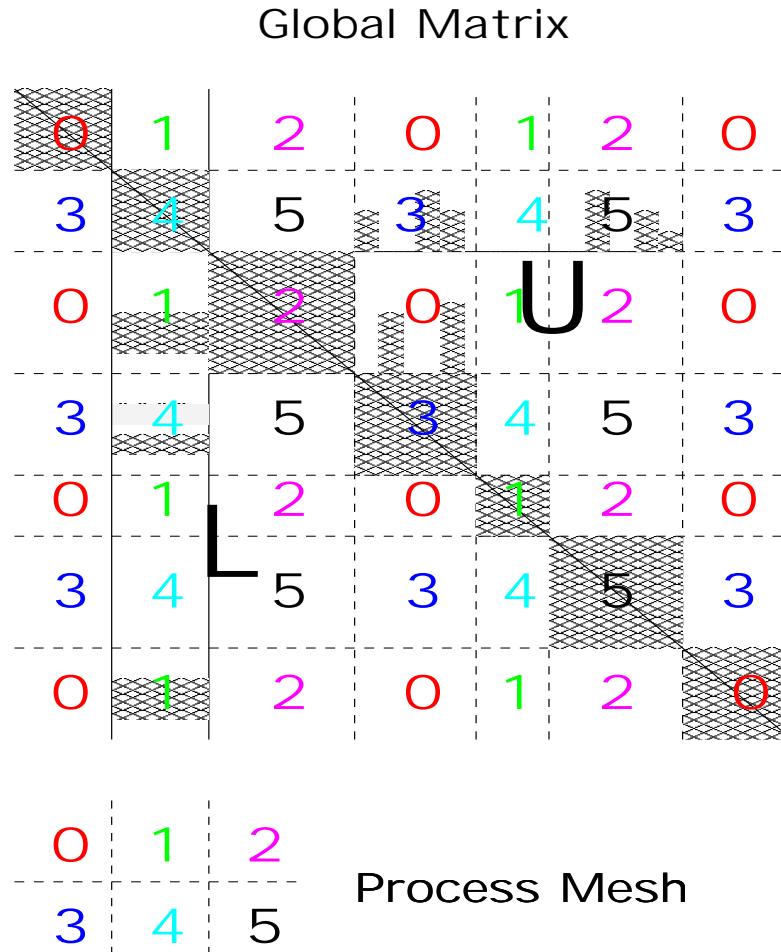


---

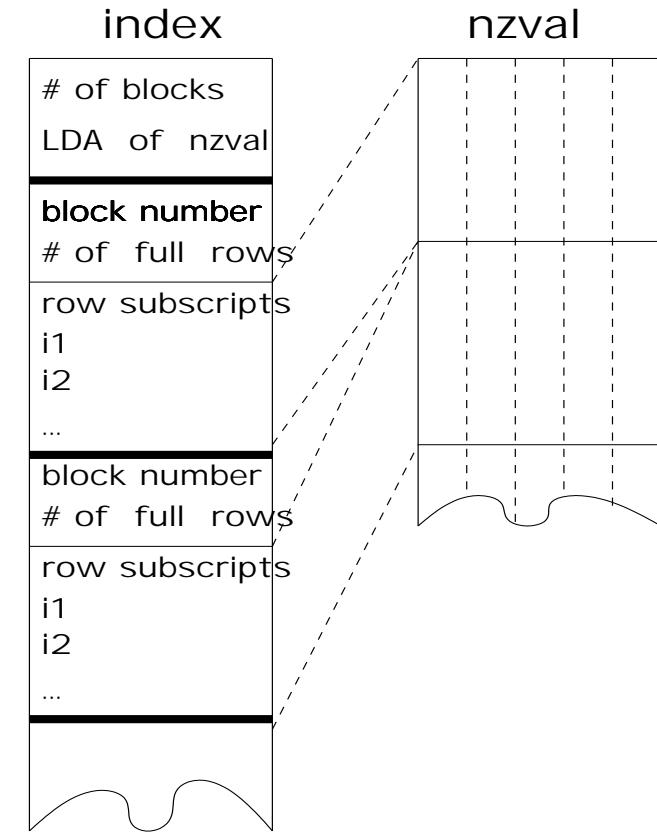
## SuperLU\_DIST – GESP Algorithm [Li/Demmel '98]

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



Storage of block column of L





## Create 2D Process Grid and MPI Communicator

- The 2D process grid/communicator must be created from an existing base MPI group/communicator. SuperLU uses the newly created communicator for all the internal communications and computations.
- Two ways to create a SuperLU process grid:
  - `superlu_gridinit(MPI_Comm Bcomm, int nprow, int ncol, gridinfo_t *grid);`  
This maps the first  $nprow * ncol$  processes in the MPI communicator Bcomm to SuperLU 2D grid.
  - `superlu_gridmap(MPI_Comm Bcomm, int nprow, int ncol,  
int usermap[], int ldumap, gridinfo_t *grid);`  
This maps an arbitrary set of  $nprow * ncol$  processes in the MPI communicator Bcomm to SuperLU 2D grid. The ranks of the selected processes are given in `usermap[]` array. For example:

	0	1	2
0	11	12	13
1	14	15	16



---

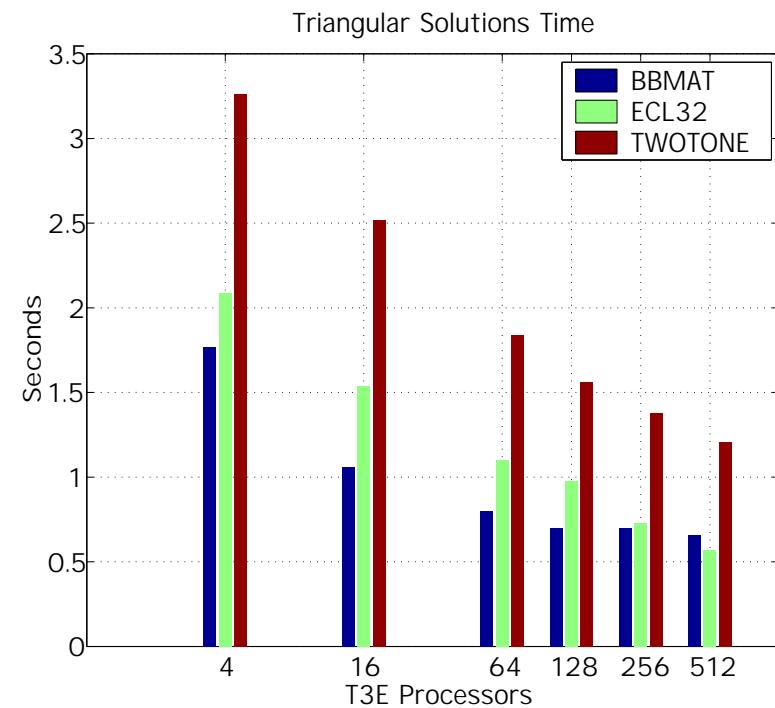
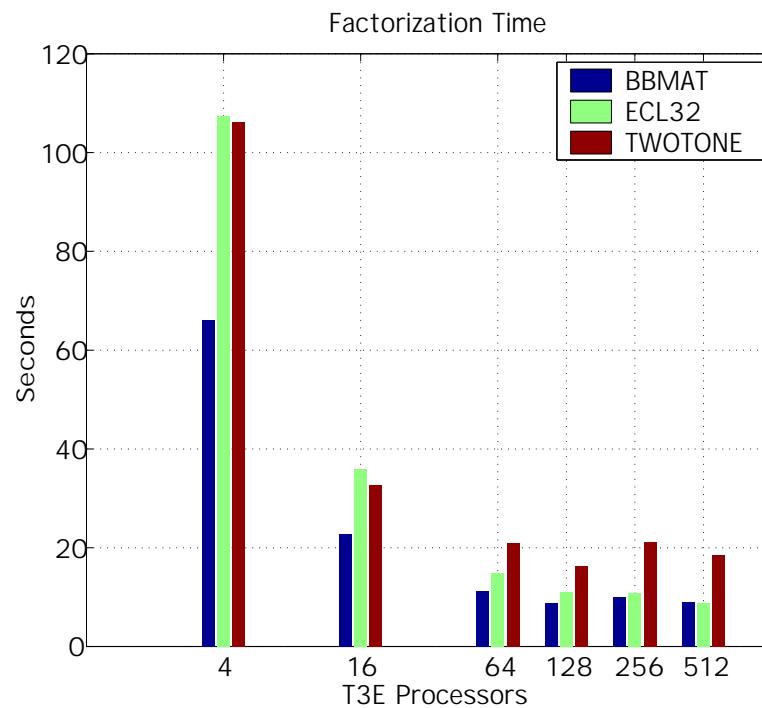
## High Performance

- *Sequential SuperLU*
  - Enhance data reuse in memory hierarchy by working with supernodes 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 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  $A^T + A$

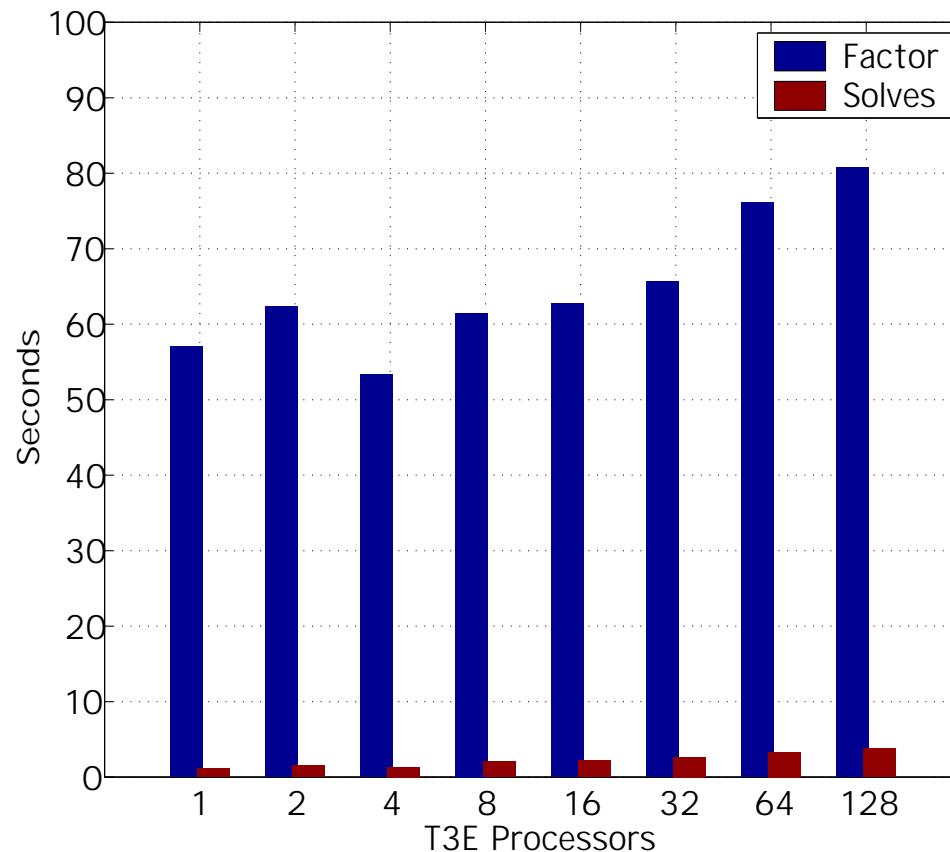
	Discipline	Symm	Order	$nnz(A)$	$nnz(L + U)$ ( $10^6$ )	Flops ( $10^9$ )
BBMAT	fluid flow	0.54	38,744	1,771,722	40.2	31.2
ECL32	device sim.	0.93	51,993	380,415	42.7	68.4
TWOTONE	circuit sim.	0.43	120,750	1,224,224	11.9	8.0



## 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\_DIST

- Study quantum scattering of 3 charged particles.  
(Baertschy, Recigno, Issacs, Li & McCurdy, LBNL/LLNL)
- Complex, unsymmetric linear systems, largest of order 8 millions.
- SuperLU\_DIST is used to build block preconditioner in a CGS solver.
- Preconditioners
  - of order 209,764, in 2 minutes, on 16 PE Cray T3E
  - of order 2 million, in 10 minutes, on 64 PE IBM SP
- These represent the first “exact” solution to a quantum mechanical 3-body Coulomb problem.
- The 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  $A^T A$  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]