Sparse Direct Solvers on High Performance Computers

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

CS267, April 24, 2000

Gaussian Elimination : Generic Algorithm

- Factorize $A = LU$, L lower triangular with unit diagonal, U upper triangular

  First step, $A = \begin{bmatrix} a & w^T \\ v & A_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/a & L_1 \end{bmatrix} \cdot \begin{bmatrix} a & w^T \\ 0 & U_1 \end{bmatrix}$

  Then, factorize $A_1 - \frac{v \cdot w^T}{a} = L_1 U_1$

- 3 nested-loops around i, j, k indices:

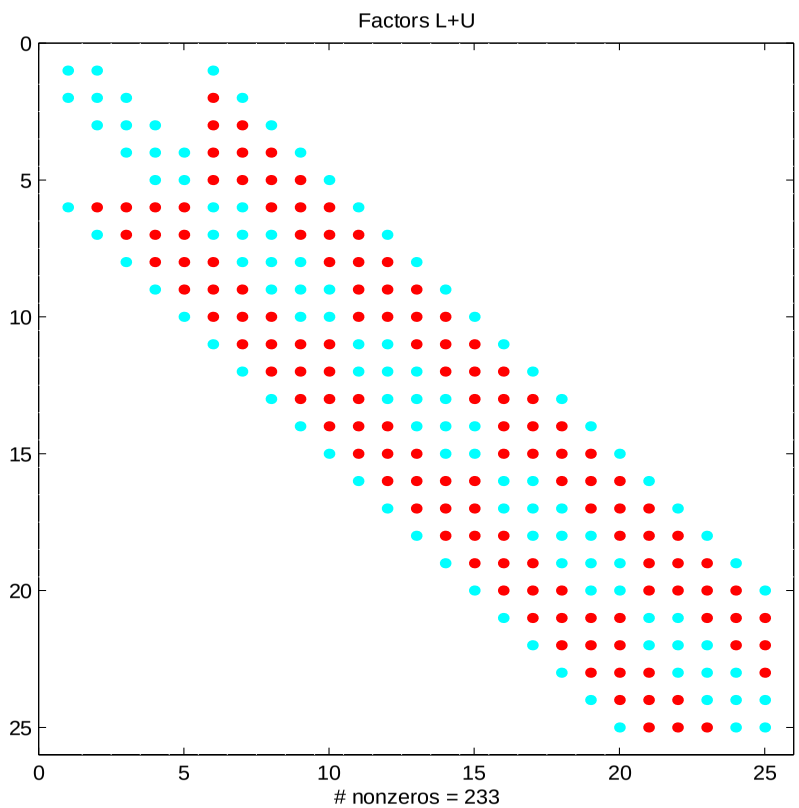
for _____ **do**
  for _____ **do**
    for _____ **do**

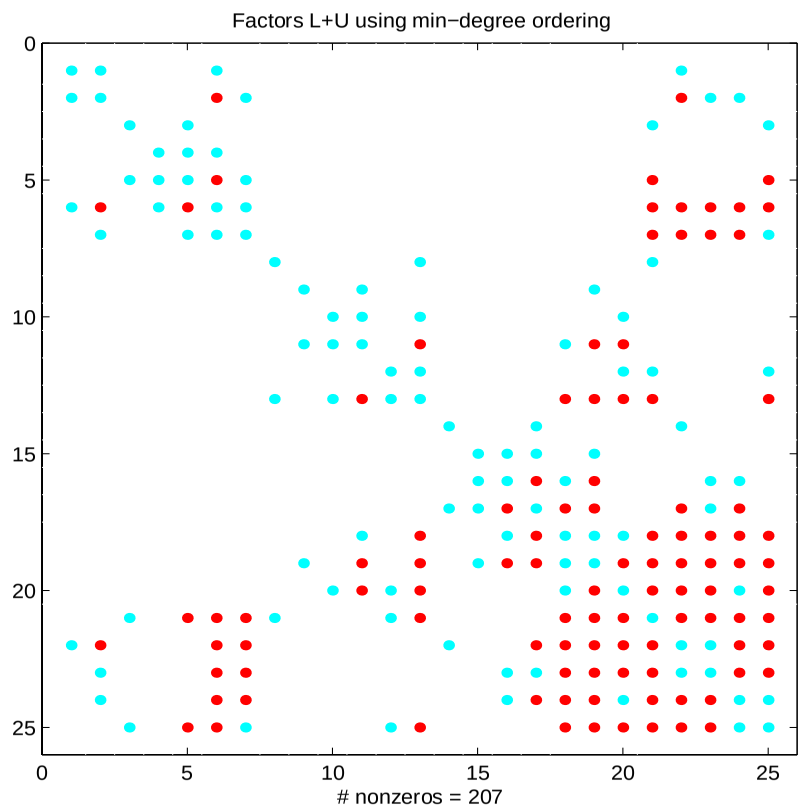
$$a_{ij} \leftarrow a_{ij} - (a_{ik} \cdot a_{kj})/a_{kk}$$

    end for;
  end for;
end for;

Sparse GE : Fill-in

- Original zero entry a_{ij} becomes nonzero after update





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. Ordering step: order equations & variables to minimize fill in L, U
     - heuristics based on combinatorics
  2. Analysis step ("Symbolic factorization"): set up data structures and allocate memory for L, U
  3. Numerical factorization
     - usually dominates total time
  4. Triangular solves
     - usually $< 5\%$ time

Ordering for Sparse Cholesky

- Minimum priority heuristics (local greedy)
  - 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]

- Graph partitioning heuristics (global)
  - 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

- Symmetric ordering for Cholesky of $A^T + A$, if pivot on diagonal
- Symmetric ordering for Cholesky of $A^T A$, if partial pivoting
  - Theorem [George/Ng '87]: If $R^T R = A^T A$ and $PA = LU$, for any P, the nonzero structure of $L + U$ is contained in that of $R^T + R$.
  - Making the upper bound 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
  - `COLMMD` in Matlab, `COLAMD` [Larimore/Davis/Gilbert/Ng '98]
- Markowitz – unsymmetric variant of minimum degree
  - Duff/Erisman/Reid '86 book
  - usually performed together with numeric factorization

Symbolic Factorization

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

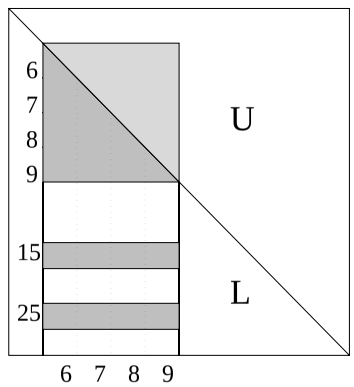
- LU
  - Use elimination graphs of L, U and their transitive reductions (elimination DAGs) [Tarjan/Rose '78, Gilbert/Liu '93, Gilbert '94]
  - Improved by symmetric structure pruning [Eisenstat/Liu '92]
  - Improved by supernodes
  - Complexity greater than $nnz(L+U)$, yet much smaller than $flops(LU)$

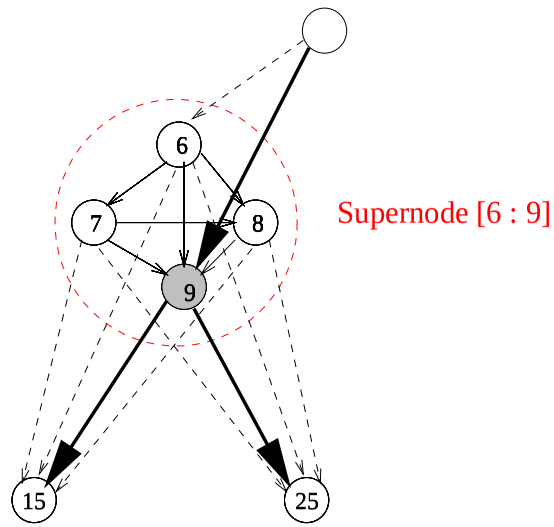
Numerical Factorization

- Usually the most expensive step ...
- 3 different approaches:
  - Submatrix-based with Markowitz ordering ['57]
  - Frontal [Irons '70] and Multifrontal [Duff/Reid '83]
  - Supernodal [e.g., SuperLU]
- Recent developments focus on
  - Superscalar processor and hierarchical memory system
  - Parallelism

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

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

- Matrix update

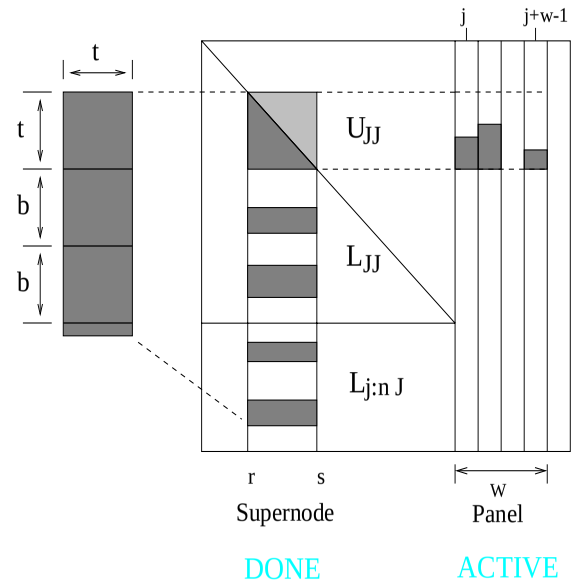
$F(s+1:n, j:j+w-1) -=$
$\quad L(s+1:n, r:s) * U(r:s, j:j+w-1)$;

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;



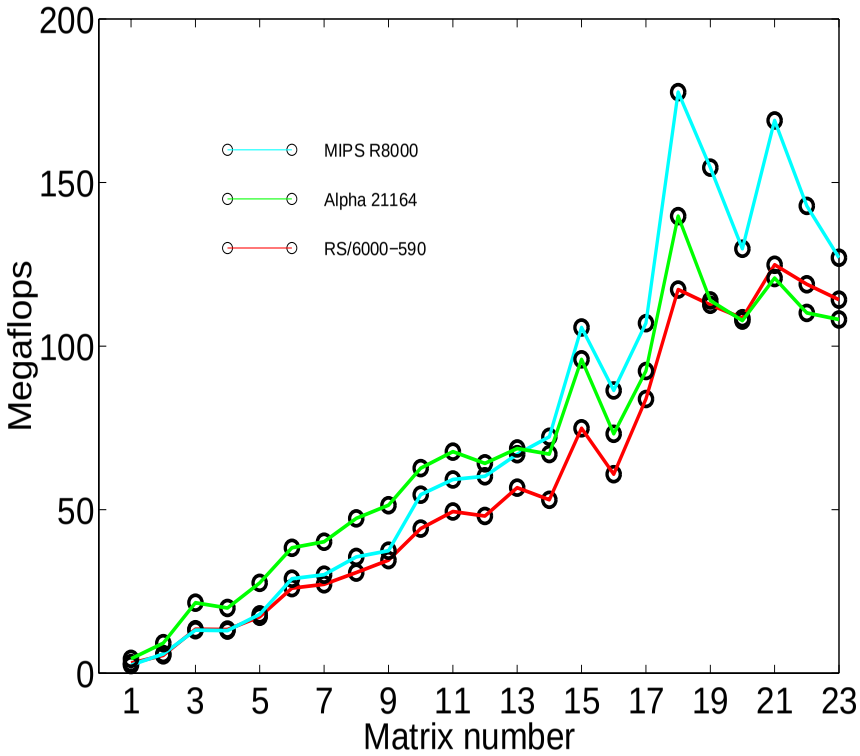
Test Matrices

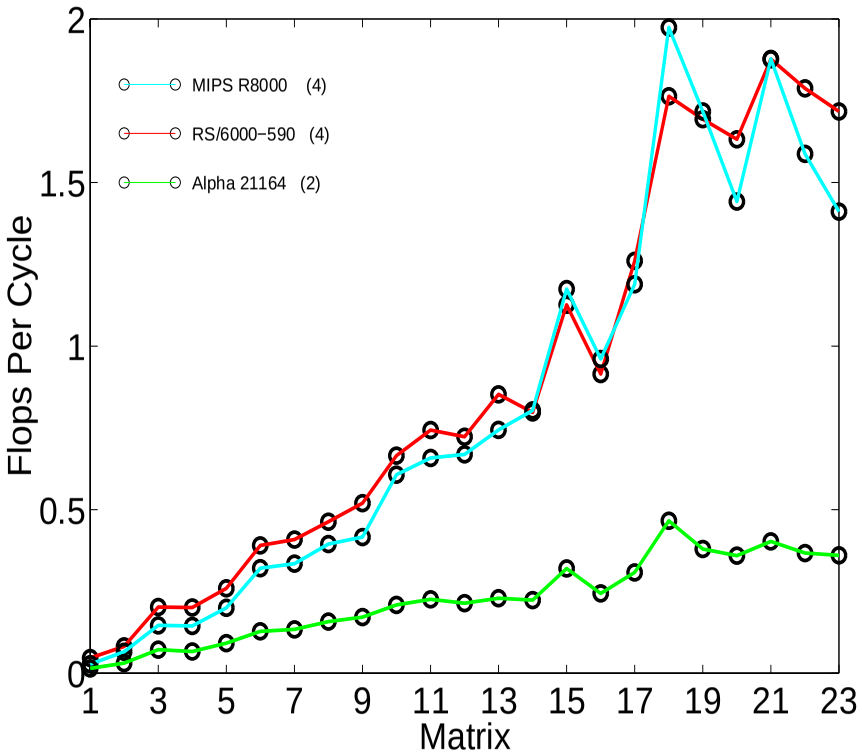
	Name	N	$nnz(A)$	$\frac{nnz(A)}{N}$	$\frac{nnz(F)}{nnz(A)}$	#flops (10^6)	$\frac{\#flops}{nnz(F)}$
1	MEMPLUS	17758	99147	5.6	1.4	1.8	12.5
2	GEMAT11	4929	33108	6.7	2.8	1.5	16.3
3	RDIST1	4134	94408	22.8	3.6	12.9	38.1
4	ORANI678	2529	90158	35.6	3.1	14.9	53.3
5	MCFE	765	24382	31.8	2.8	4.1	59.9
6	LNSP3937	3937	25407	6.5	16.8	38.9	91.1
7	LNS3937	3937	25407	6.5	17.7	44.8	99.7
8	SHERMAN5	3312	20793	6.3	12.0	25.2	101.3
9	JPWH991	991	6027	6.1	23.4	18.0	127.7
10	SHERMAN3	5005	20033	4.0	21.6	60.6	139.8
11	ORSREG1	2205	14133	6.4	28.5	59.8	148.6
12	SAYLR4	3564	22316	6.3	29.3	104.8	160.0
13	SHYY161	76480	329762	4.3	23.2	1571.6	205.8
14	GOODWIN	7320	324772	44.4	9.6	665.1	213.9
15	VENKAT01	62424	1717792	27.5	7.6	3219.9	247.9
16	INACCURA	16146	1015156	62.9	9.8	4118.7	414.3
17	AF23560	23560	460598	19.6	30.4	6363.7	454.9
18	DENSE1000	1000	1000000	1000	1.0	666.2	666.2
19	RAEFSKY3	21200	1488768	70.2	11.8	12118.7	690.7
20	EX11	16614	1096948	66.0	23.8	26814.5	1023.1
21	WANG3	26064	177168	6.8	74.9	14557.5	1095.5
22	RAEFSKY4	19779	1316789	66.6	20.3	31283.4	1172.6
23	VAVASIS3	41029	1683902	41.0	29.2	89209.3	1813.5

SuperLU Uniprocessor Factorization Rates

[Demmel/Eisenstat/Gilbert/Li/Liu '95]

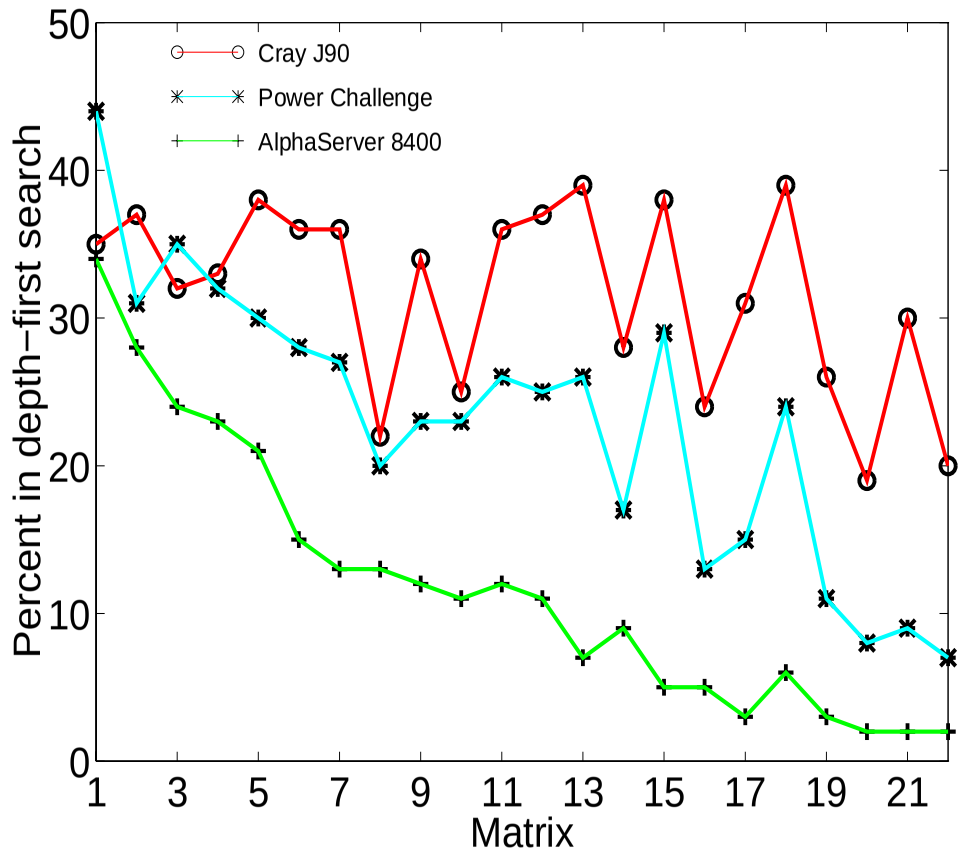
- Time includes everything except column ordering





Fraction of Time in Symbolic Factorization on 1-CPU

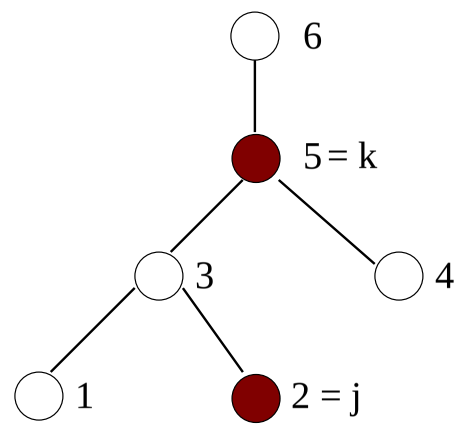
- Show relative strength of integer vs. floating-point speed
- Roughly carry over to shared memory parallel code



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 $nnz(A)$.

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



Parallel Task Scheduling for SMPs [Demmel/Gilbert/Li '97]

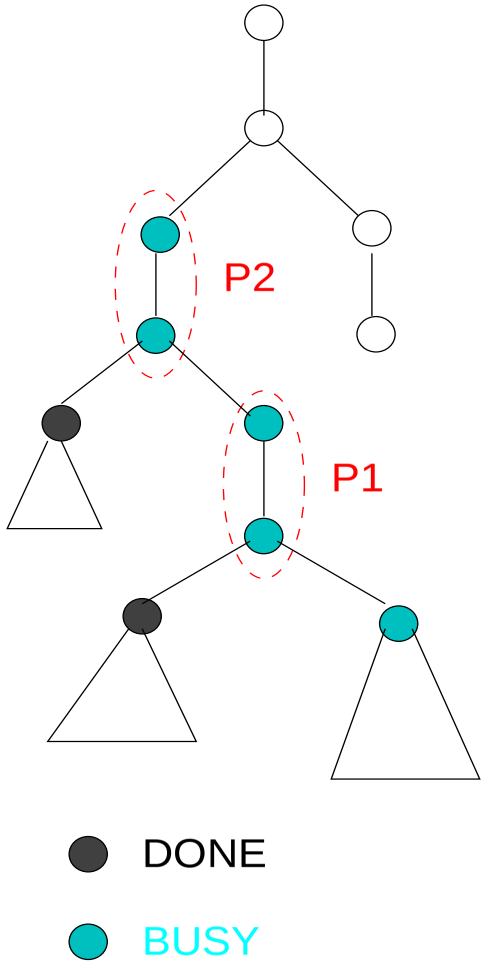
Shared task queue initialized with leaves;

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_MT Factorization Speed

- Time, Mflops and speedup on Origin 2000

Matrix	N	$nnz(A)$	P=1	P=8	P=18	Speedup
EX11	16614	1096948	209	33	20	10
RAEFSKY4	19779	1316789	229	39	25	9
BBMAT	38744	1771722	605	166	64	9
VAVASIS3	41092	1683902	598	136	75	8
TWOTONE	120750	1224224	313	58	38	8

- A 3-D flow calculation (EX11)

Machine	CPUs	Speedup	Mflops	Percent peak
C90	8	6	2583	33%
J90	16	12	831	25%
Power Challenge	12	7	1002	23%
Origin 2000	20	10	1335	17%
AlphaServer 8400	8	7	781	17%

SuperLU_MT Factorization Speed

	Matrix	nnz(L+U) (10^6)	Flops (10^9)	Seconds P=1	Seconds P=4	Seconds P=8	Speedup	Mflops	Percent Peak
Cray C90	AF23560	14.0	6.4	36.2	9.4	6.2	4.9	1035	13%
	EX11	26.2	26.8	75.4	21.2	10.4	6.5	2538	33%
	RAEFSKY4	26.7	31.3	78.3	21.0	13.1	5.5	2399	31%
AlphaServer 8400	Matrix	nnz(L+U) (10^6)	Flops (10^9)	Seconds P=1	Seconds P=4	Seconds P=8	Speedup	Mflops	Percent Peak
	AF23560	14.0	6.4	70.8	18.1	11.6	5.8	553	12%
	EX11	26.2	26.8	245.3	64.6	34.2	7.1	781	16%
	RAEFSKY4	26.7	31.3	288.2	74.1	42.8	6.6	734	15%
Origin2000	Matrix	nnz(L+U) (10^6)	Flops (10^9)	Seconds P=1	Seconds P=8	Seconds P=18	Speedup	Mflops	Percent Peak
	EX11	26.2	26.8	209.6	33.1	20.3	10.0	1335	19%
	RAEFSKY4	26.7	31.3	229.5	39.2	25.7	9.0	1222	17%
	BBMAT	50.4	45.5	605.3	166.3	64.1	9.0	710	10%
	TWOTNE	23.9	12.5	313.4	57.9	39.3	8.0	318	5%

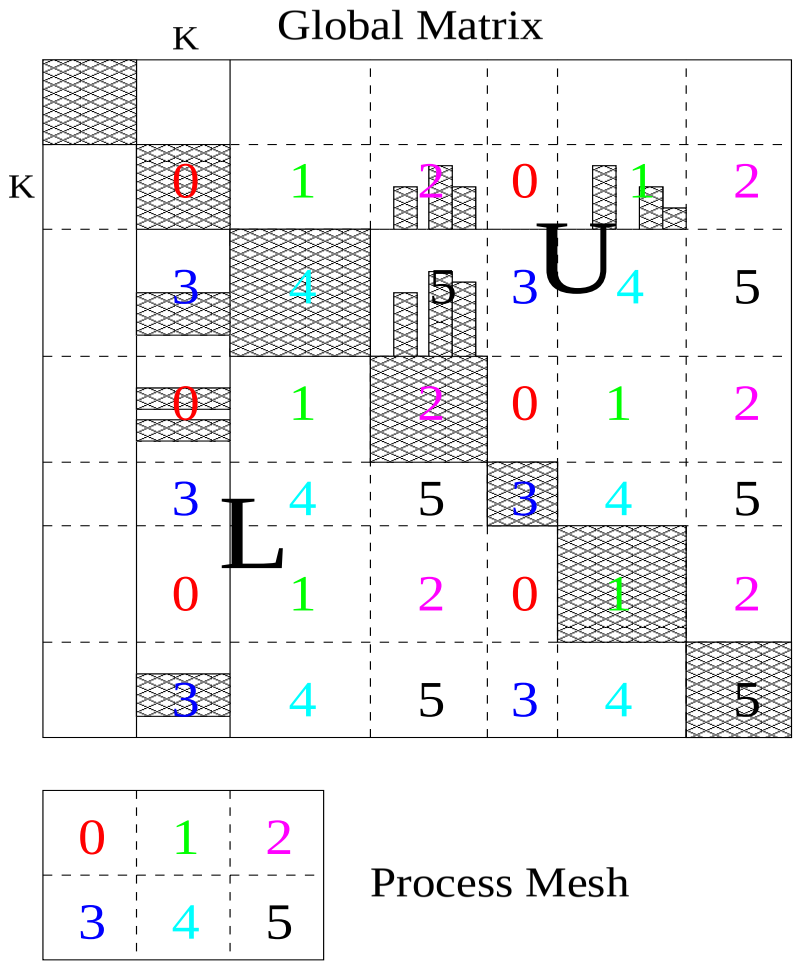
SuperLU_Dist – for Distributed Memory Machines

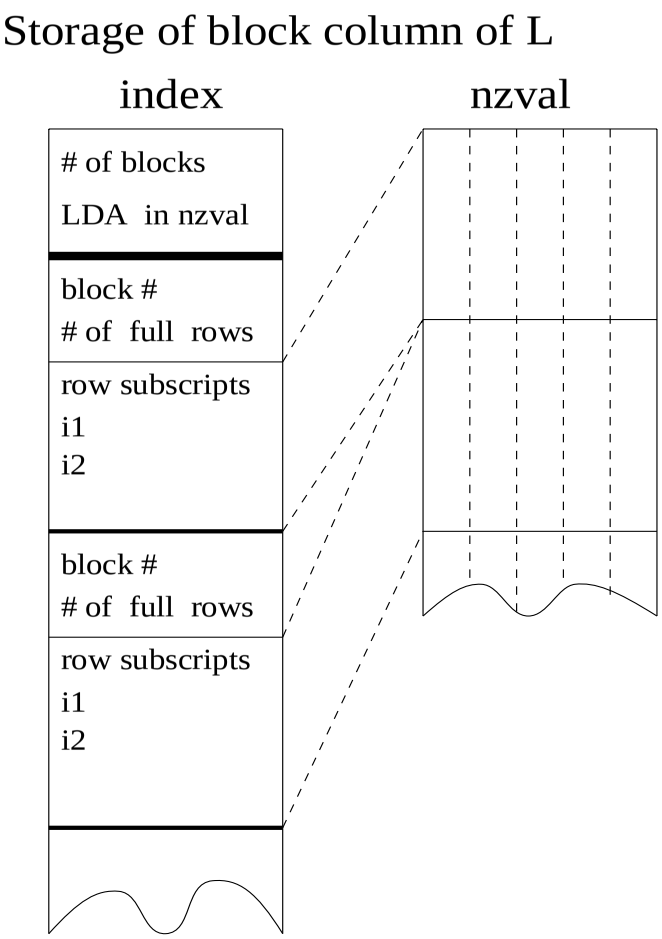
- The SMP algorithm in SuperLU_MT is hard to scale because
  - Not exploit enough parallelism
  - Many fine-grained messages due to dynamic scheduling, load balancing and adaption of data structures

- Algorithmic changes for scalability
  - Parallelize the loops over both rows and columns (2-D block-cyclic)
  - Static Pivoting (GESP)
    - Pivot before numerical factorization so data structures static
    - Accommodate possible pivot growth during factorization without changing data structures

    $\Longrightarrow$ symbolic/numerical algorithms decoupled

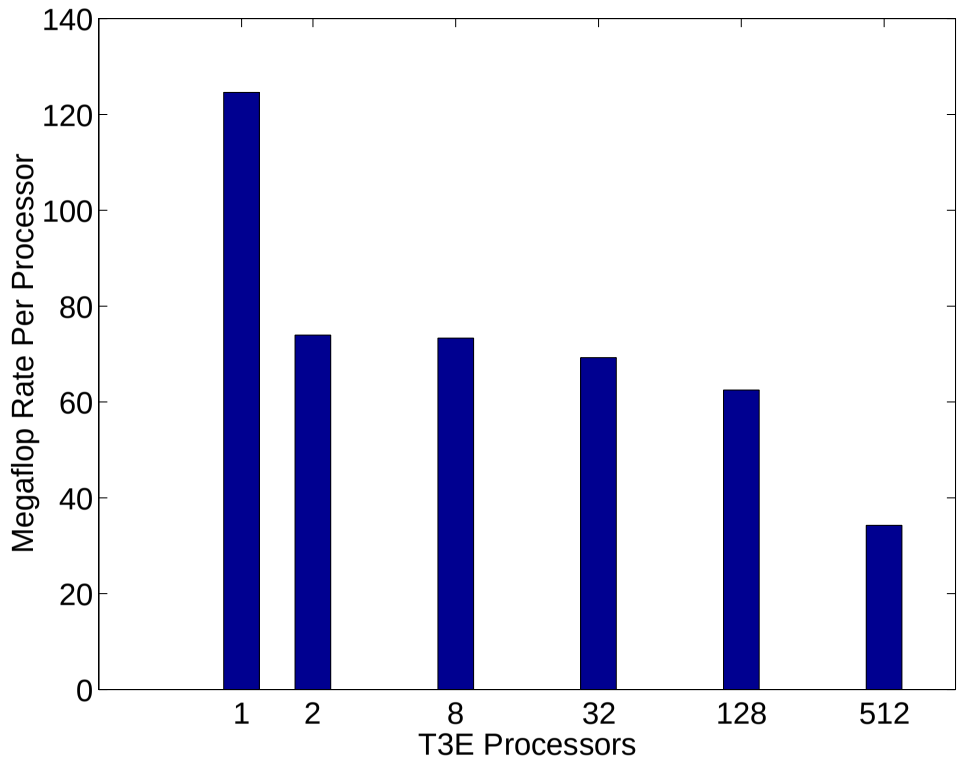
Distributed Data Structure: 2D Block Cyclic Layout





Scaling on T3E

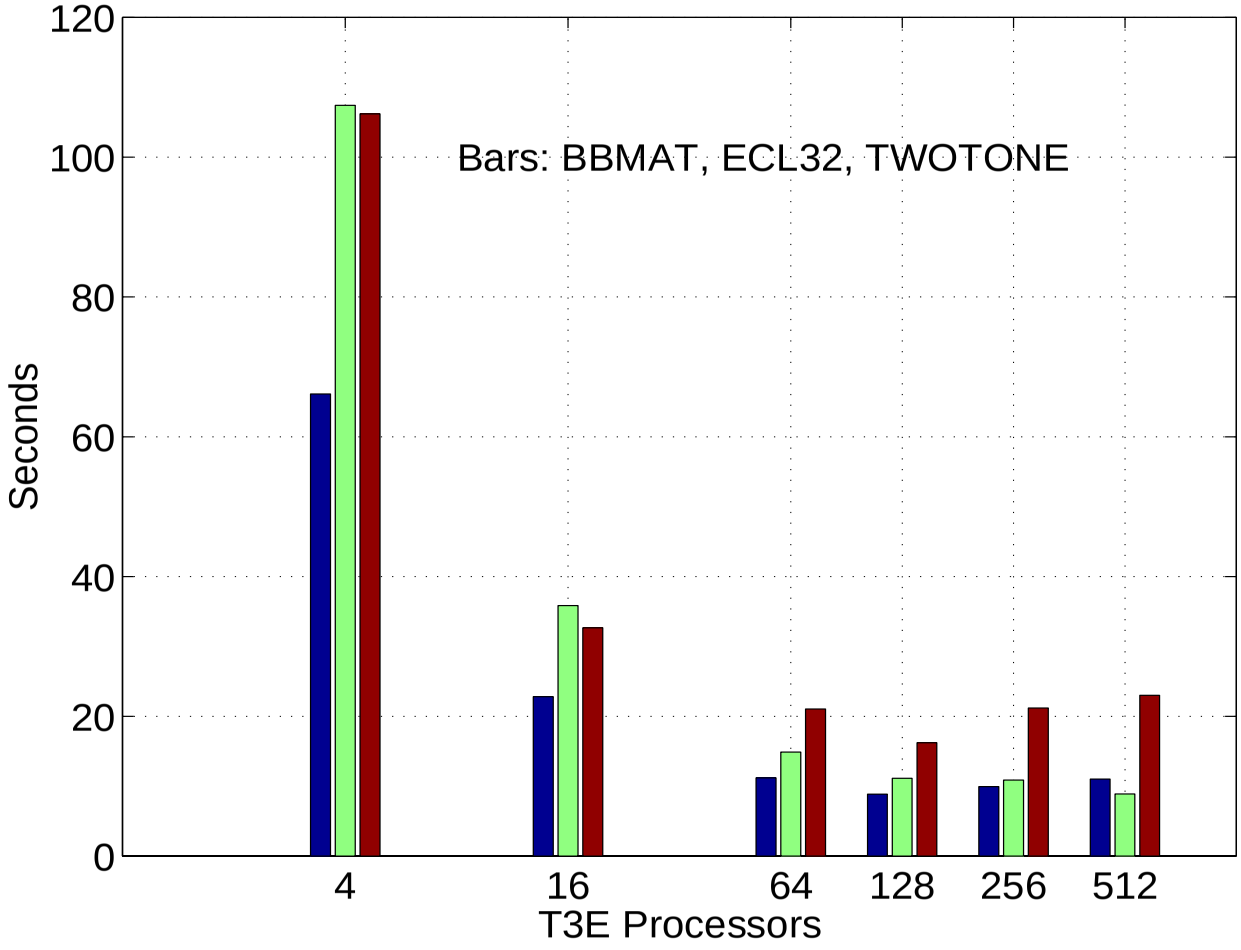
- 3D Grids, 11 points stencil
- Grid size increases with number of PEs, such that Flops per PE roughly constant



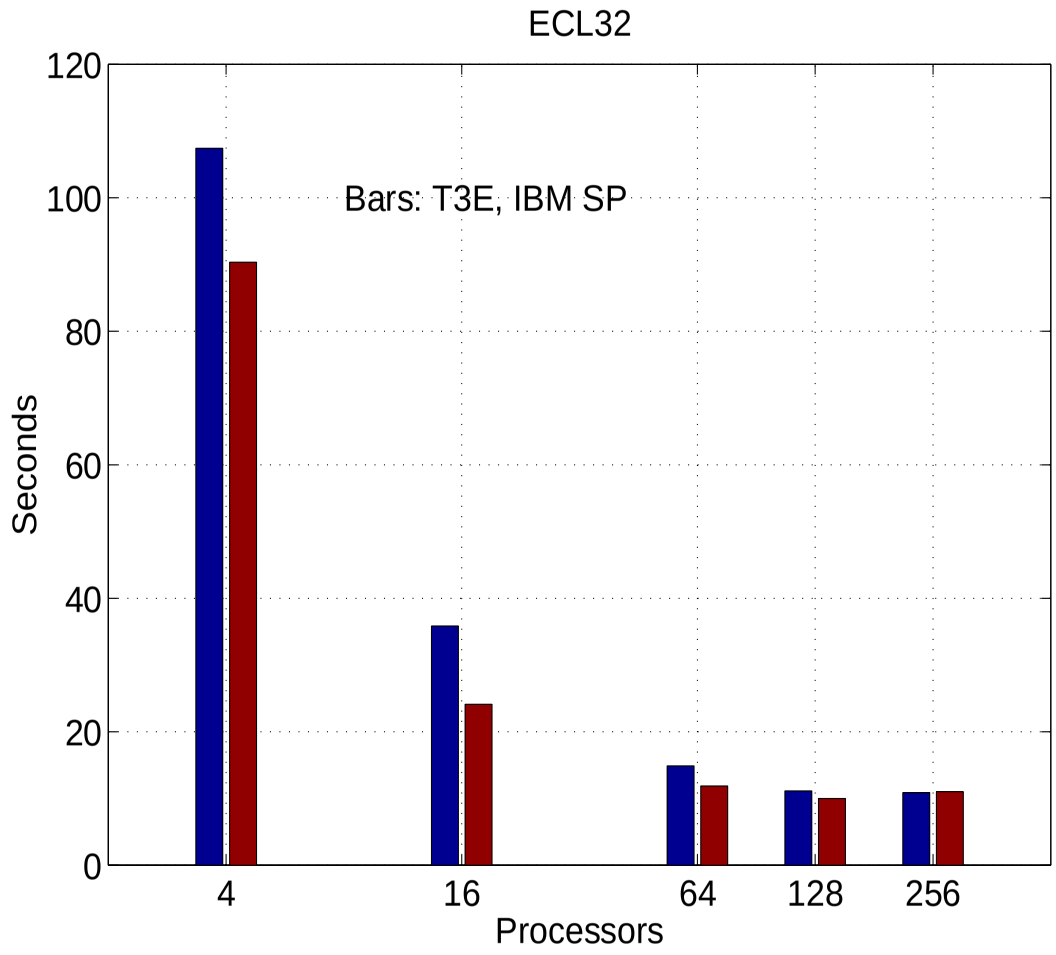
Irregular Problems

Matrix	Discipline	Symmetry	N	nnz(A)	nnz(L+U) (10^6)	Flops (10^9)
BBMAT	CFD	0.54	38,744	1,771,722	35.9	26.6
ECL32	Device Sim	0.93	51,993	380,415	41.4	60.6
TWOTONE	Circuit Sim	0.43	120,750	1,224,224	10.7	7.3

Irregular Problems: Time on T3E



Irregular Problem : Time on T3E and IBM SP



Ordering Impact on Scalability : Matrix ECL32

- Time on T3E

Ordering	nnz(L+U) (10^6)	P=4	P=32	P=128	P=512	Gflops/s
MMD(A'*A)	73.5	325.0	60.5	21.5	14.3	8.5
MMD(A'+A)	41.4	107.4	20.6	11.1	8.9	6.8
ND(A'+A)	24.3	49.0	12.0	8.7	9.6	2.2