A Scalable Sparse Direct Solver Using Static Pivoting

Xiaoye S. Li
xiaoye@nersc.gov
NERSC, Lawrence Berkeley National Laboratory

James W. Demmel
demmel@cs.berkeley.edu
University of California at Berkeley

Outline

- Motivation
- Numerical results of static pre-pivoting vs. partial pivoting
- Distributed data structures, algorithms, and performance
- Conclusion and future work

Review: Supernode-Panel Sparse GEPP

- Supernodal algorithm [Demmel/Eisenstat/Gilbert/Li/Liu '95]
  - Exploit memory hierarchy by panel and 2-D blocking; Enable BLAS 2 or 3
  - Reduce symbolic time by traversing coarser graph, and symmetric structure reductions [Eisenstat/Liu '92]
- Barrier-free algorithm on SMPs [Demmel/Gilbert/Li '97]
  - Use column elimination tree to exploit two levels of concurrency
  - Processors dynamically obtain tasks through a central task queue
- Available: SuperLU and SuperLU_MT
  - www.nersc.gov/~xiaoye
  - www.netlib.org/scalapack/prototype

Motivation of Our New Approach

- GEPP is harder to scale up than Cholesky factorization
- Computational graph does not unfold until run-time due to numerical pivoting
  - Requires dynamic adaptation of data structures
  - Symbolic algorithm is not separable from numerical algorithm
  - Requires fine-grained communication to perform dynamic scheduling and load balancing
- Our approach is not pivot in the usual way, thereby enabling the static optimizations that make Cholesky scalable

Outline of GESP Algorithm

(1) Row/column equilibration and row permutation: $A \leftarrow P_r \cdot D_r \cdot A \cdot D_c$, where D_r and D_c are diagonal matrices and P_r is a row permutation chosen to make the diagonal large compared to the off-diagonal

(2) Find a column permutation P_c to preserve sparsity: $A \leftarrow P_c \cdot A \cdot P_c^T$
For example, minimum degree ordering on $A^T A$

(3) Factorize $A = L \cdot U$ with control of diagonal magnitude

if ($|a_{ii}| < \sqrt{\varepsilon} \cdot ||A||$) **then**
  set a_{ii} to $sign(a_{ii})\sqrt{\varepsilon} \cdot ||A||$
endif

(4) Solve $A \cdot x = b$ using the L and U factors, followed by iterative refinement [Arioli/Demmel/Duff '89]

Row Permutation P_r for Large Diagonal

- A is represented as an undirected weighted bipartite graph
- Solve a weighted bipartite matching problem on this graph
- Can maximize different properties of the diagonal of P_rA, such as
  - The smallest magnitude
  - The sum of the magnitudes
  - The product of the magnitudes
- We use the algorithm by [Duff/Koster '98]
  Maximizing the product of the diagonal magnitudes
  - Space complexity $O(nnz(A))$
  - Worst case time complexity $O(n \cdot nnz(A) \cdot \log n)$

Iterative Refinement

- Newton's method applied to $f(x) = b - A \cdot x$

iterate:

$r = b - A \cdot x$ . . . sparse matrix-vector multiply

Solve $A \cdot dx = r$ . . . triangular solve

$berr = \max_i \frac{|r|_i}{(|A| \cdot |x| + |b|)_i}$ . . . componentwise backward error

if ($berr > \varepsilon$ and $berr \leq \frac{1}{2} \cdot lastberr$) **then**

$x = x + dx$

$lastberr = berr$

goto iterate

endif

- 0 – 3 steps usually suffice

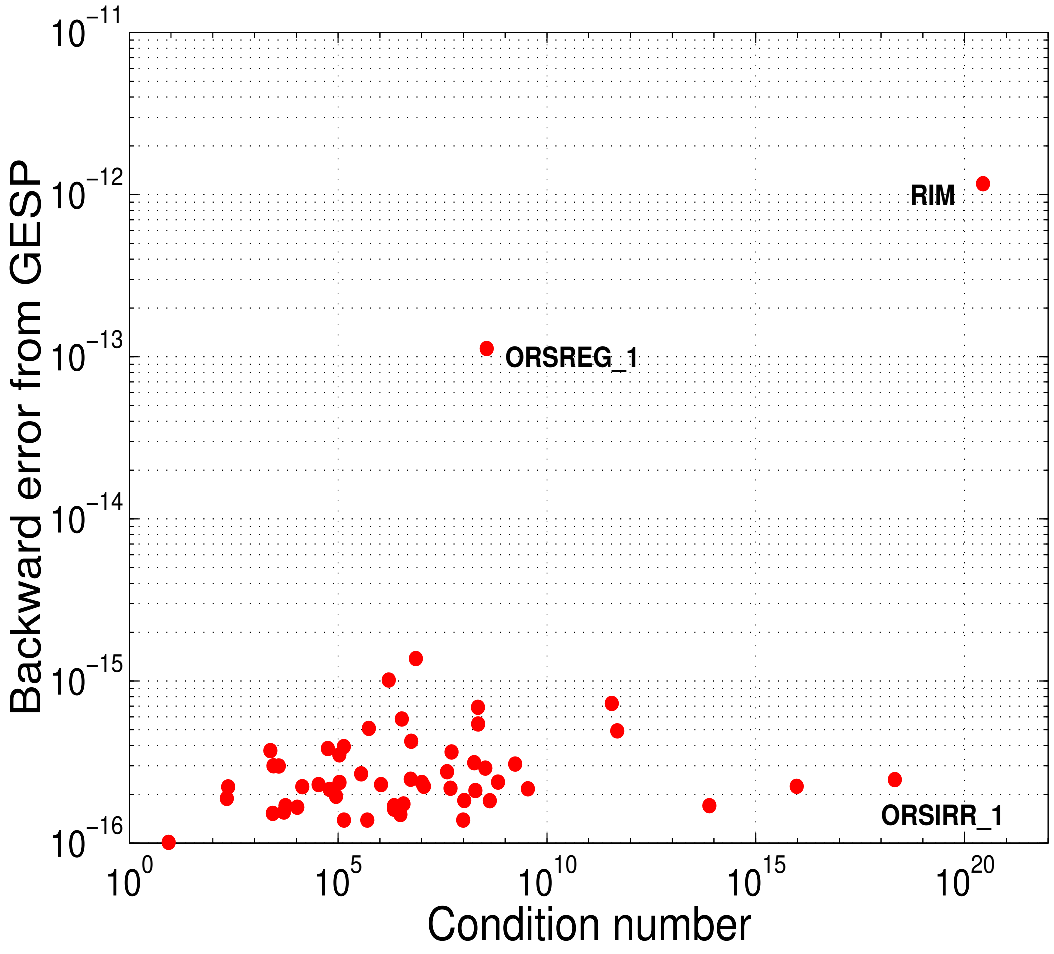
Numerical Results of GESP

- 55 unsymmetric matrices
  - 22 contain zeros on diagonal to begin with which remain zero
  - 5 create zeros on diagonal during elimination
    (garon2, jpwh_991, orsirr_1, orsreg_1, psmig_1)
  - Most of the other 28 would get large errors due to pivot growth
- Working precision is double (64 bits)
- Detailed results in tabular form at www.nersc.gov/~xiaoye/SuperLU

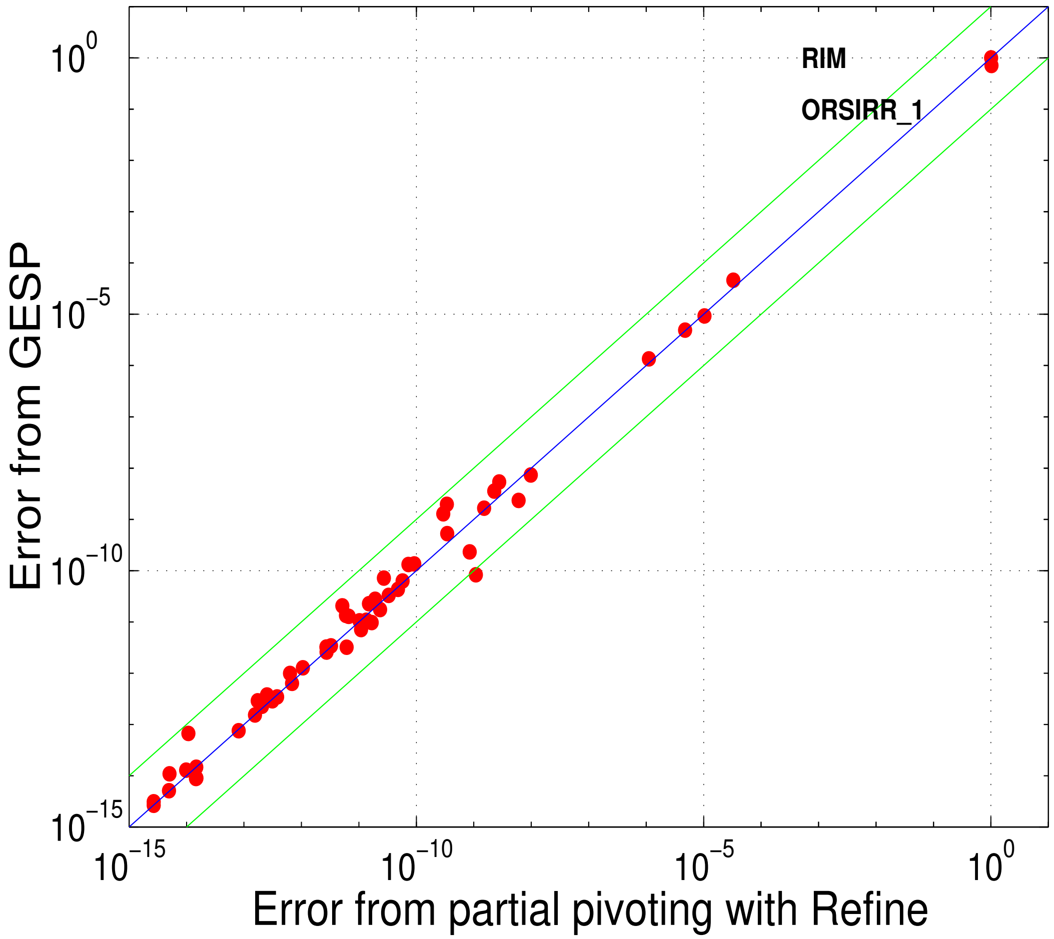
Application Domains

Discipline	Matrices
fluid flow, CFD	af23560, bbmat, bramley1, bramley2, ex11, fidapm11, garon2, graham1, inv-extrusion-1, lnsp3937, lns_3937, mixing-tank, raefsky3, rma10, venkat01, wu
fluid mechanics	goodwin, rim
circuit simulation	add32, gre_1107, jpwh_991, memplus, onetone1, onetone2, twotone
device simulation	wang3, wang4, ecl32
chemical engineering	extr1, hydr1, lhr01, radfr1, rdist1, rdist2, rdist3a, west2021
petroleum engineering	orsirr_1, orsreg_1, sherman3, sherman4, sherman5
finite element PDE	av4408, av11924
stiff ODE	fs_541_2
Olmstead flow model	olm5000
aeroelasticity	tols4000
reservoir modelling	pores_2
crystal growth simulation	cry10000
power flow modelling	gemat11
dielectric waveguide	dw8192 (eigenproblem)
astrophysics	mcfe
plasma physics	utm5940
demography	psmigr_1
economics	mahindas, orani678

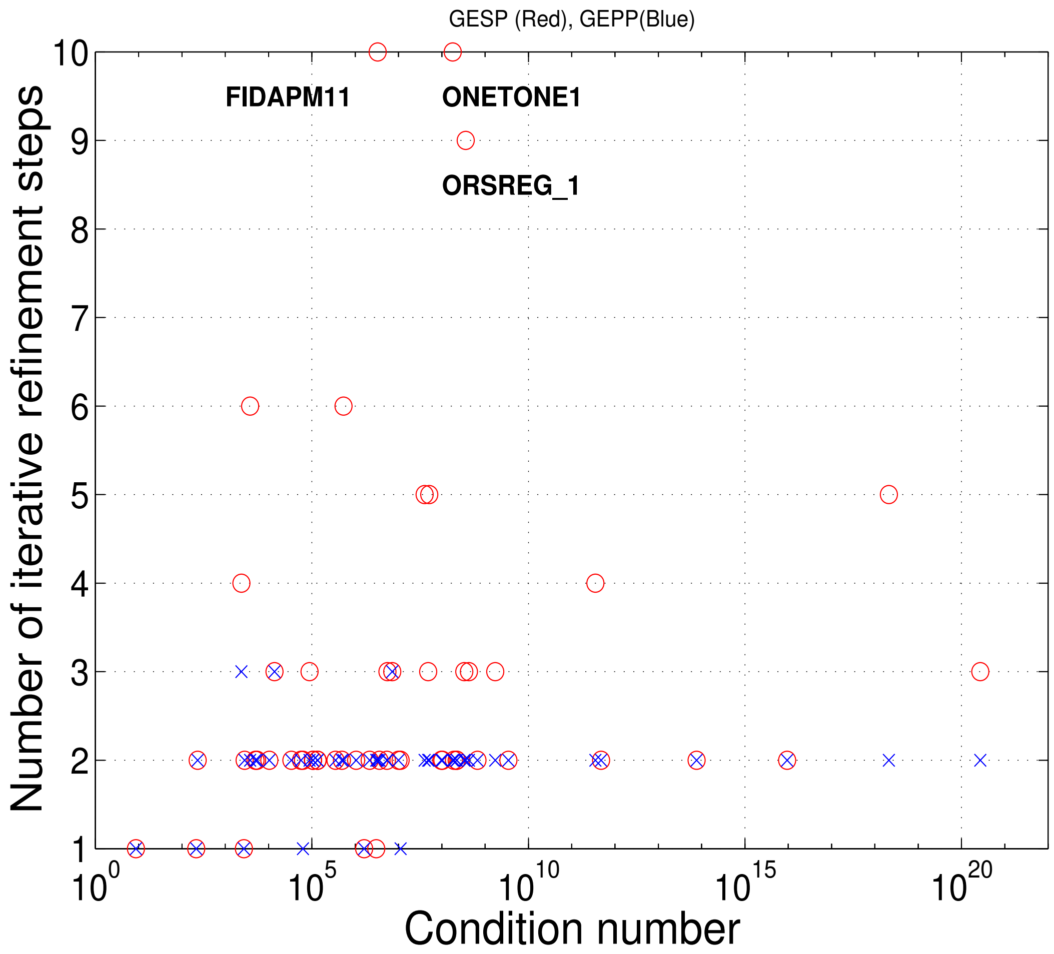
The Sparse Backward Error $\max_i \frac{|A\cdot x - b|_i}{(|A|\cdot|x|+|b|)_i}$



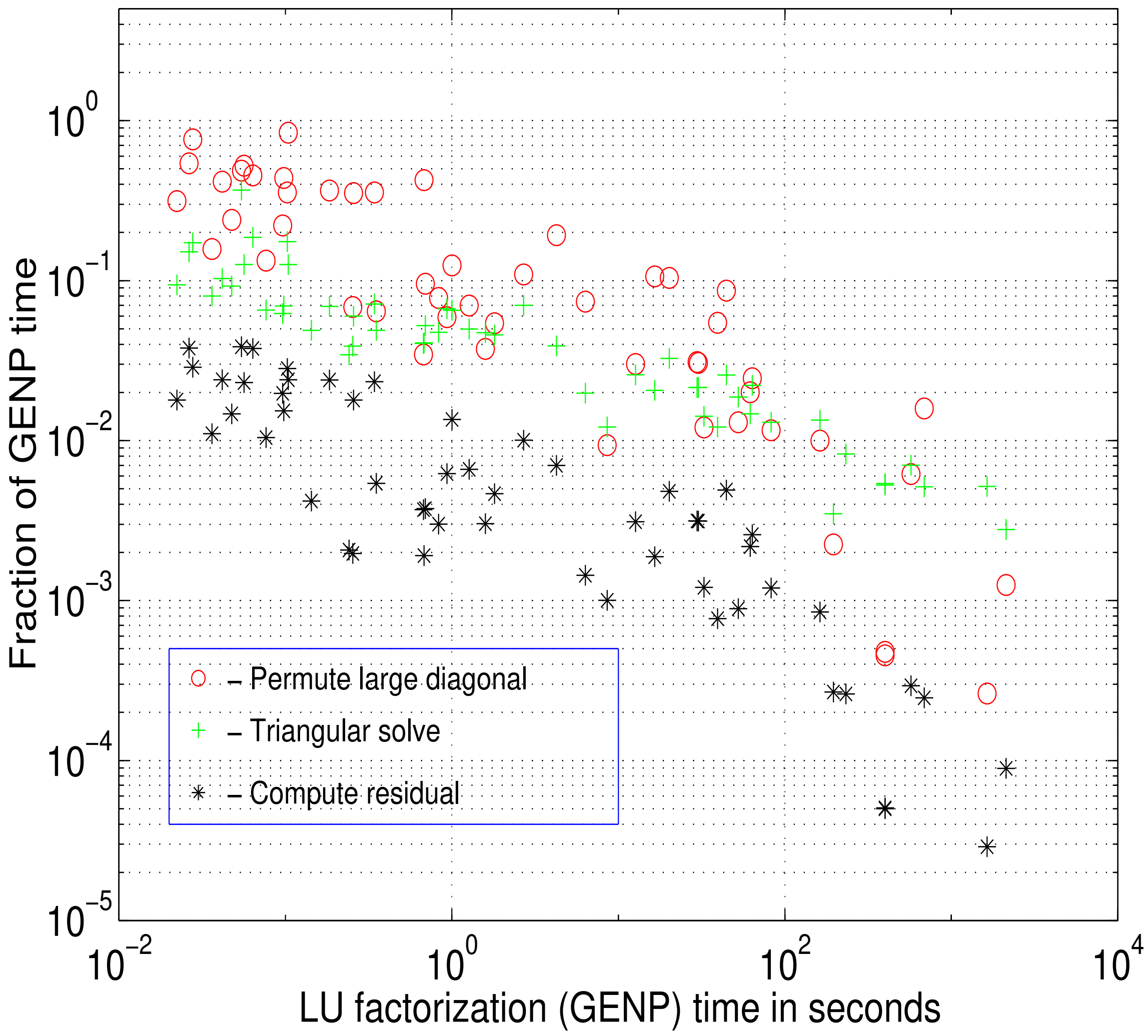
The Error $\frac{||x_{true}-x||_\infty}{||x||_\infty}$



Iterative Refinement Steps



Runtime Compared With LU Factorization



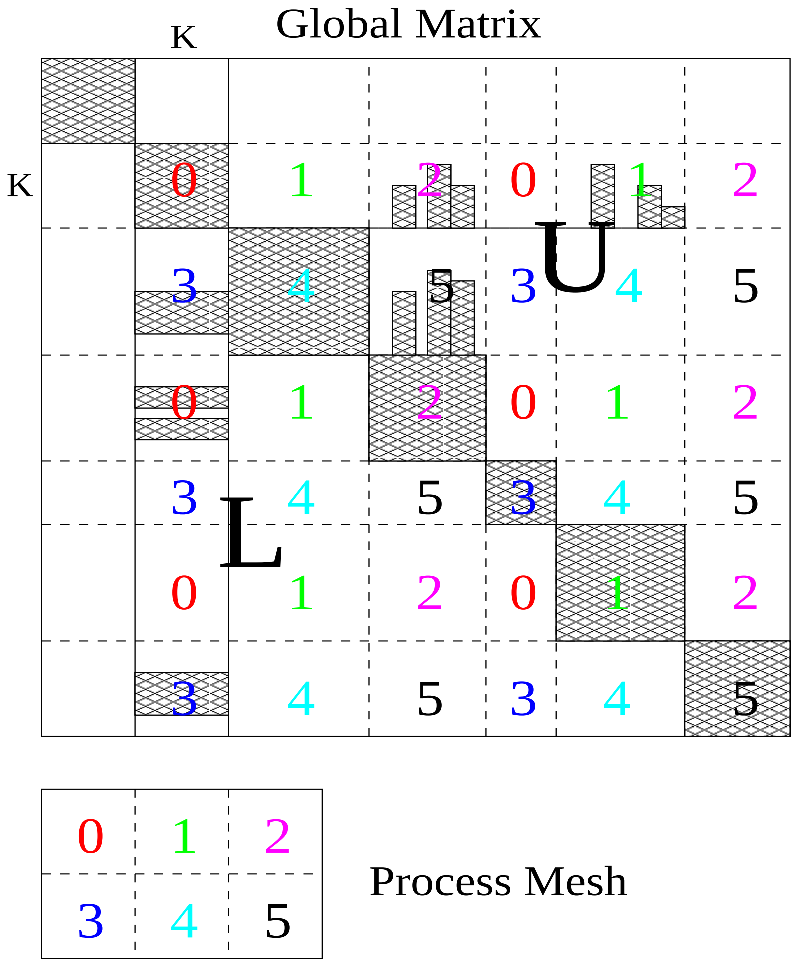
Scaling Analysis

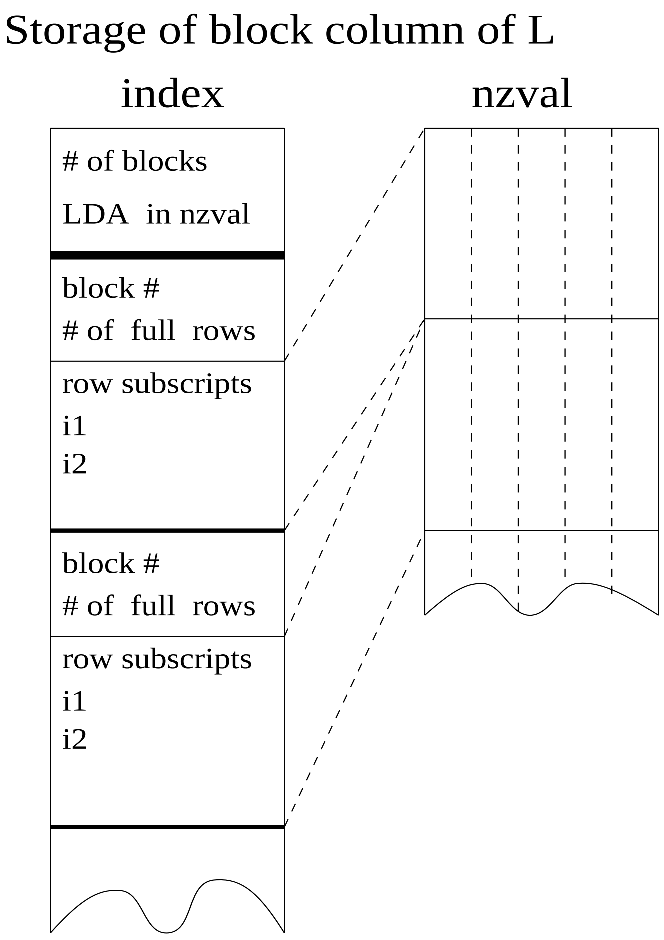
Example: 2-D $k \times k$ regular grid, with nested dissection ordering

- Cholesky factorization $A = LL^T$ [George/Liu '81]
  - Nonzeros in L: $O(k^2 \log k)$
  - Floating-point operations: $O(k^3)$
- Communication volume
  - 1-D mapping: $\Omega(k^2 P)$ [George/Liu/Ng '89]
  - 2-D mapping: $\Omega(k^2 \sqrt{P} \log P)$ [Rothberg/Gupta '93]
- Observed cross-over point between 1-D and 2-D: $P \leq 32$

Distributed Data Structure

Block $A(I, J)$ maps to process $(I \bmod NPROW, J \bmod NPCOL)$





Distributed Right-looking LU Using MPI

for block $K = 1$ **to** N **do**

(1) **if** ($mycol = PROC_C(K)$) **then**

Obtain block column factor of $L(K:N, K)$

Send $L(K:N, K)$ to the processes in my row who need it

else

Receive $L(K:N, K)$ from processes $PROC_C(K)$ if I need it

endif

(2) **if** ($myrow = PROC_R(K)$) **then**

Parallel triangular solves : $U(K, K+1:N) = L(K, K)^{-1} \cdot A(K, K+1:N)$

Send $U(K, K+1:N)$ to processes in my column who need it

else

Receive $U(K, K+1:N)$ from processes $PROC_R(K)$ if I need it

endif

(3) **for** $J = K+1$ to N **do**

for $I = K+1$ to N **do**

if ($myrow = PROC_R(I)$ & $mycol = PROC_C(J)$ & $L(I, K) \neq 0$ & $U(K, J) \neq 0$) **then**

$A(I, J) = A(I, J) - L(I, K) \cdot U(K, J)$

endif

end for

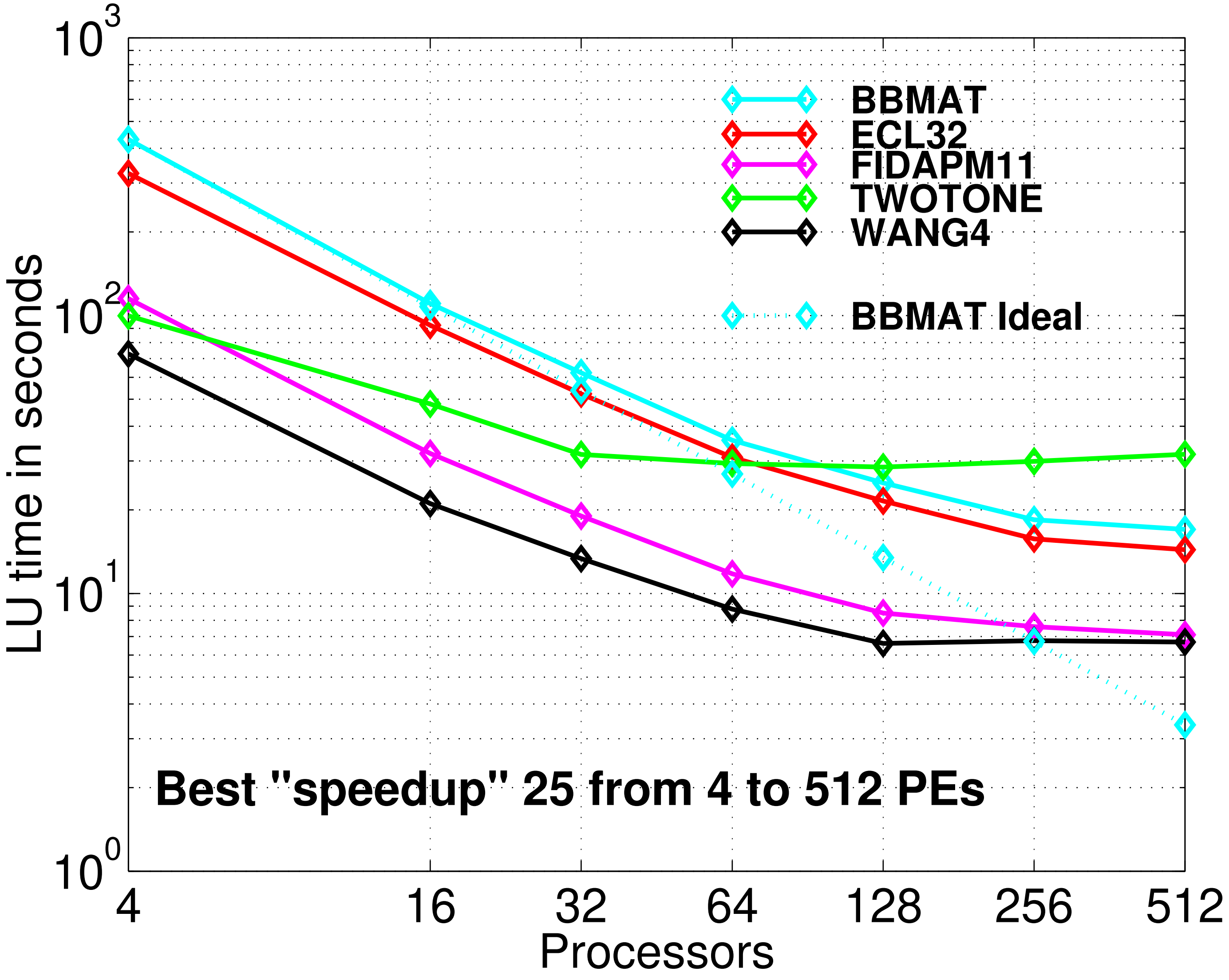
Parallel Performance

- Portable implementation with MPI
  Tested on Cray T3E, ASCI Blue, IBM SP2, Berkeley NOW

- Matrices and their best performance on 512 node T3E-900

	Discipline	Order	$nnz(A)$	$nnz(L+U)$ (10^6)	Flops (10^9)	Mflops
BBMAT	Fluid flow	38744	1771722	49.1	4.3	2493
ECL32	Device sim.	51993	380415	73.5	120.4	8419
FIDAPM11	Fluid flow	22294	623554	23.0	17.9	2291
TWOTONE	Circuit sim.	120750	1224224	22.6	8.7	297
WANG4	Device sim.	26068	177196	27.7	35.3	5542

LU Time on Cray T3E-900

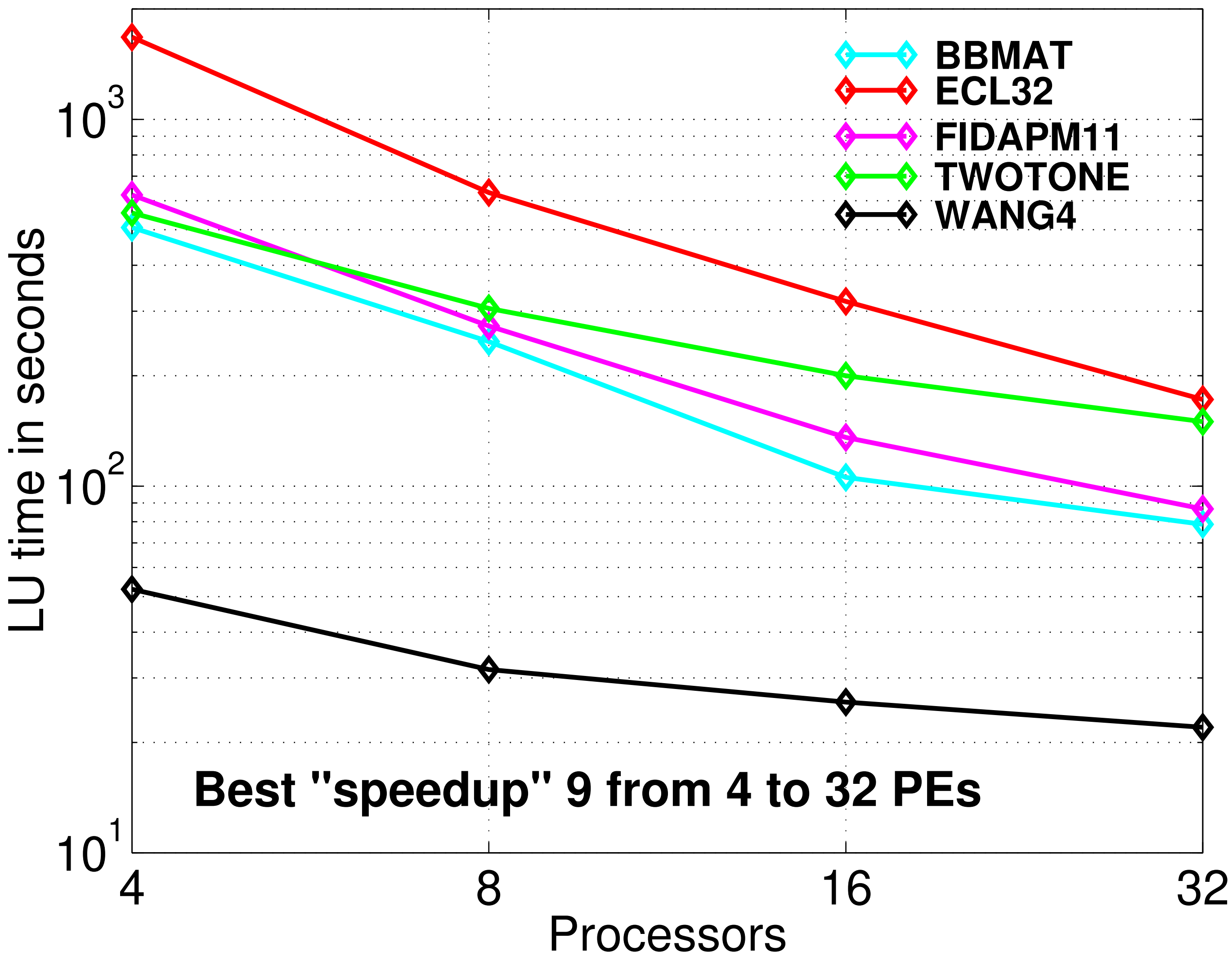


LU Time on Cray T3E-900

	Symbolic	Numeric							
		P=4	16	32	64	128	256	512	Mflops
BBMAT	11.8	430.8	110.5	62.3	35.6	25.8	18.4	17.0	2493
ECL32	14.0	325.0	92.3	52.2	30.8	21.5	15.7	14.3	8419
FIDAPM11	4.1	115.1	31.9	19.0	11.77	8.5	7.6	7.1	2291
TWOTONE	6.6	99.8	48.1	31.6	29.4	28.5	29.9	31.7	297
WANG4	4.2	72.8	21.1	13.3	8.8	6.6	6.76	6.59	5542

- Best "speedup" 25, from 4 to 512 processors

LU Time On IBM SP2



LU Time on IBM SP2

	P=4	8	16	32	Mflops
BBMAT	506.9	247.8	105.6	78.7	710
ECL32	1675.8	631.9	319.0	172.4	678
FIDAPM11	622.6	273.4	135.8	86.9	187
TWOTONE	555.8	305.1	205.2	158.3	69
WANG4	52.3	31.6	25.8	22.0	436

- Best “speedup” 9, from 4 to 32 processors

Distributed Lower Triangular Solve $L \cdot x = b$

$lsum = 0$; $x = b$;

for each K that I own   ... Compute leaf nodes

    if ($myrow = PROC_R(K)$ & $mycol = PROC_C(K)$ & $frecv[K] = 0$)

        $x(K) = L(K,K)^{-1} \cdot x(K)$

        Send $x(K)$ to processes in column $PROC_C(K)$

    endif

while (I have more work) **do**   ... Compute internal nodes

    Receive a message

    if (message is $lsum(K)$)

        $x(K) = x(K) + lsum(K)$;

        if ($--frecv(K) = 0$)

            $x(K) = L(K,K)^{-1} \cdot x(K)$

            Send $x(K)$ to the column processes $PROC_C(K)$

    else if (message is $x(K)$)

        for each $I > K$, $L(I,K) \neq 0$ that I own

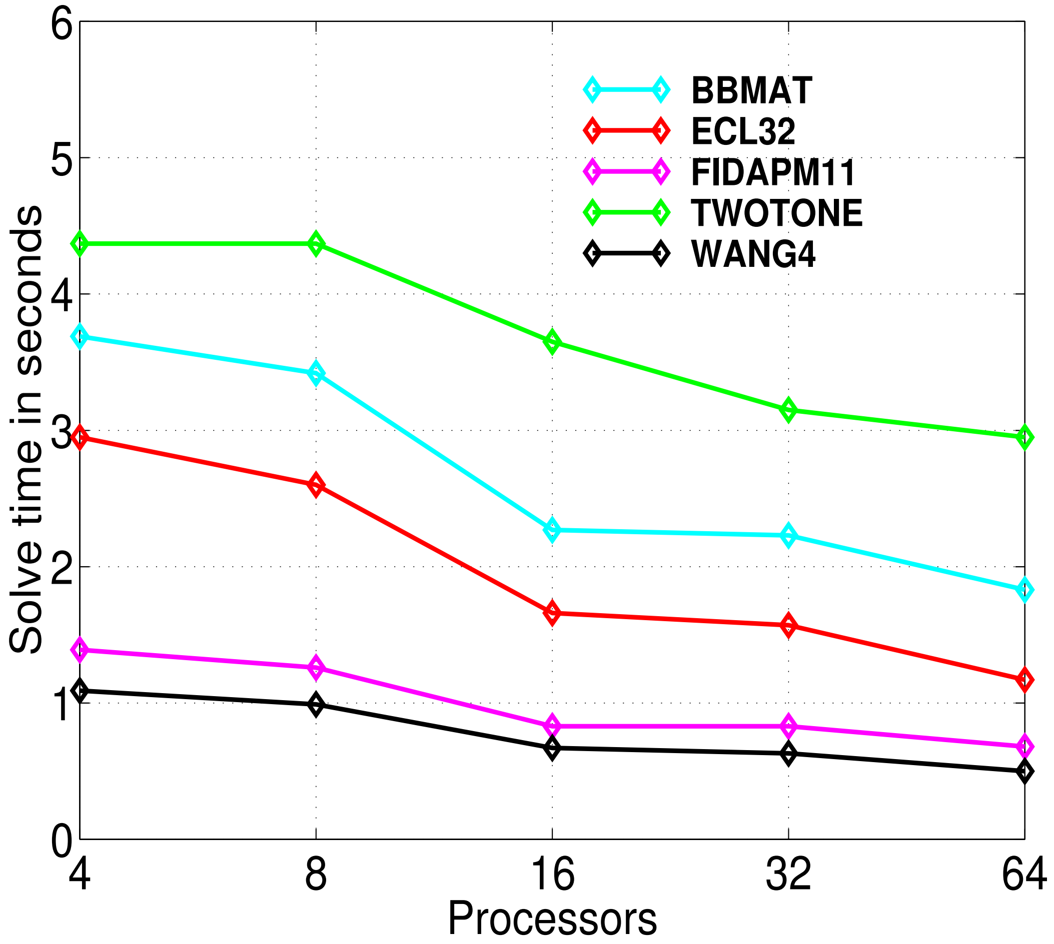
            $lsum(I) = lsum(I) - L(I,K) \cdot x(K)$

            if ($--fmod(I) = 0$)

                Send $lsum(I)$ to the diagonal process who owns $L(I,I)$

    endif

Triangular Solve (One Right-hand Side) Time on T3E-900



- Fast: 1% to 3% of the LU factorization time on 4 processors
- More than 95% of time in communication

Load Balance and Communication on 64 PE T3E

Load balance factor $B = \frac{\Sigma_i(f_i)}{P \max_i(f_i)}$.

	AF23560	BBMAT	ECL32	EX11	FIDAPM11	RMA10	TWOTONE	WANG4
B_{fact}	.82	.77	.94	.87	.70	.73	.43	.92
B_{sol}	.84	.81	.92	.83	.81	.76	.66	.88
Comm								
$fact$	.82	.54	.54	.77	.59	.92	.92	.62
sol	.97	.97	.96	.97	.97	.96	.96	.97

One Application

- Quantum chemistry: Calculate electron impact ionization cross-sections [Baertschy/McCurdy, LBNL]

- Complex, non-Hermitian systems
  - Of order 209,764 on 16 PE Cray T3E, in 2 minutes
  - Of order 736,164 on 24 PE ASCI Blue, in 21 minutes

Conclusions and Future Work

- Conclusions
  - Novel ideas to avoid partial pivoting, and numerically as stable as partial pivoting for wide range of problems
  - Fine-tuned distributed data structure enables scalability

- Future
  - More techniques to control element growth
    - Low rank modification to A with Sherman-Morrison-Woodbury fixup
    - Store some entries and compute internally with extra precision
  - Higher performance
    - Use EDAGs to improve matrix layout and eliminate false dependency in the computation
    - Switch to dense code in the end
  - Incomplete factorization

Future Work

- More techniques to control element growth
  - Low rank modification to A with Sherman-Morrison fixup
  - Store some entries and compute internally with extra precision

- Higher performance
  - Use EDAGs to improve matrix layout and eliminate false dependency in the computation
  - Switch to dense code in the end

- Incomplete factorization