Efficient Sparse LU Factorization with Lazy Space Allocation

Bin Jiang, Steven Richman, Kai Shen and Tao Yang
Department of Computer Science
University of California
Santa Barbara, CA 93106
http://www.cs.ucsb.edu/research/S+

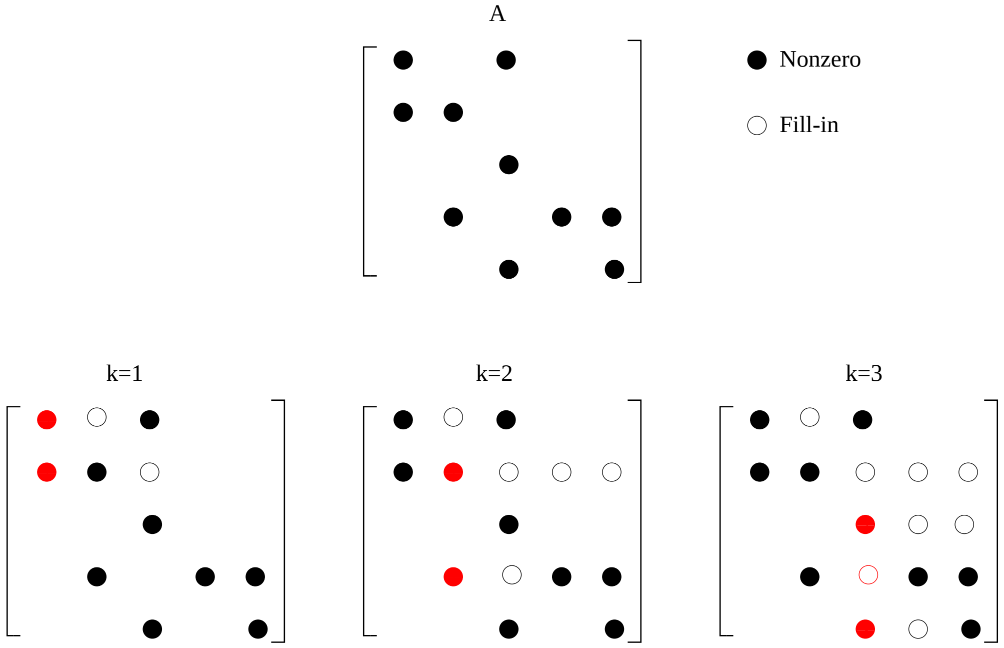
Preliminaries

- Gaussian elimination with partial pivoting: used for solving a linear system $Ax = b$ $(PA = LU)$.
- Main tasks: Factor(k), Update(k,j)
- Challenges for parallel sparse LU factorization:
  - Poor caching performance due to irregular data distribution.
  - Limited parallelism due to pivoting.
  - High space cost for asynchronous computation.
- Related work: SuperLU, UMFPACK, S^*, S^+.

Static symbolic factorization

Purpose: Avoid dynamic space allocation at runtime.

Idea: Statically predict all possible fill-ins and compute the structural bound for L and U factors. [George and Ng, 1987]



Problem: Excessive over-estimation of fill-ins for certain matrices.

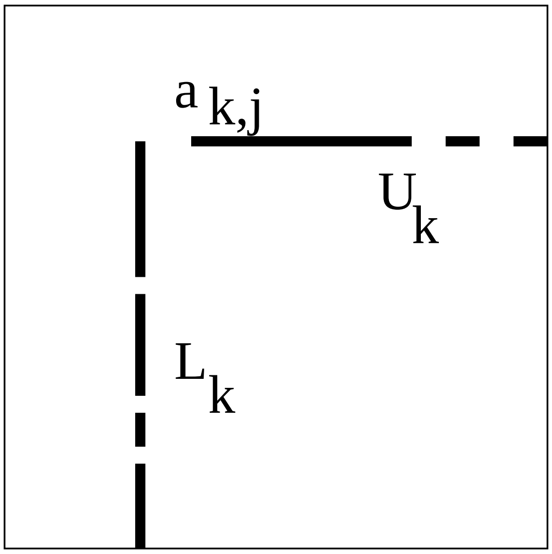
$\Longrightarrow$ Extra computation

Fast Sparse LU code: S^+

Techniques to alleviate the impact of extra computation:

- Elimination forest guided supernode partitioning and amalgamation to identify dense structures.
- Elimination forest guided 2D asynchronous computation scheduling. (DAG representation has more overhead.)
- Supernodal matrix multiplication GEMM to improve the kernel computation speed.
- Lazy space allocation to control memory usage.

Elimination forest: Definition

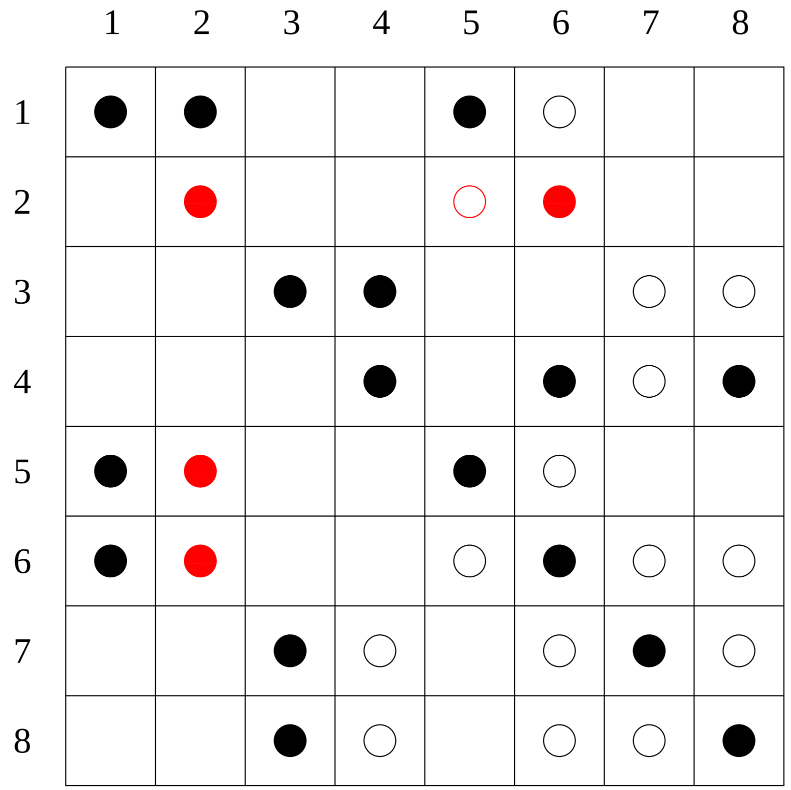


- $a_{k,j}$ — a matrix element after symbolic factorization.
- U_k — k-th row at U part.
- L_k — k-th column at L part.

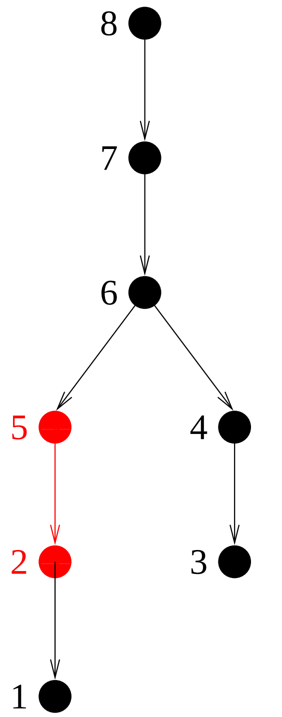
Elimination forest (E-forest):

- Each node corresponds to a column.
- There is an edge from j to k if a_{kj} is the first off-diagonal nonzero in U_k, and $|L_k| > 1$.

Example



Elimination Forest

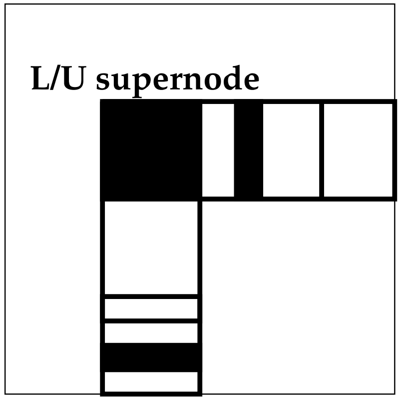


● Nonzeros in the original matrix

○ Fill-in entries generated by symbolic factorization

Use E-forest in L/U supernode partitioning

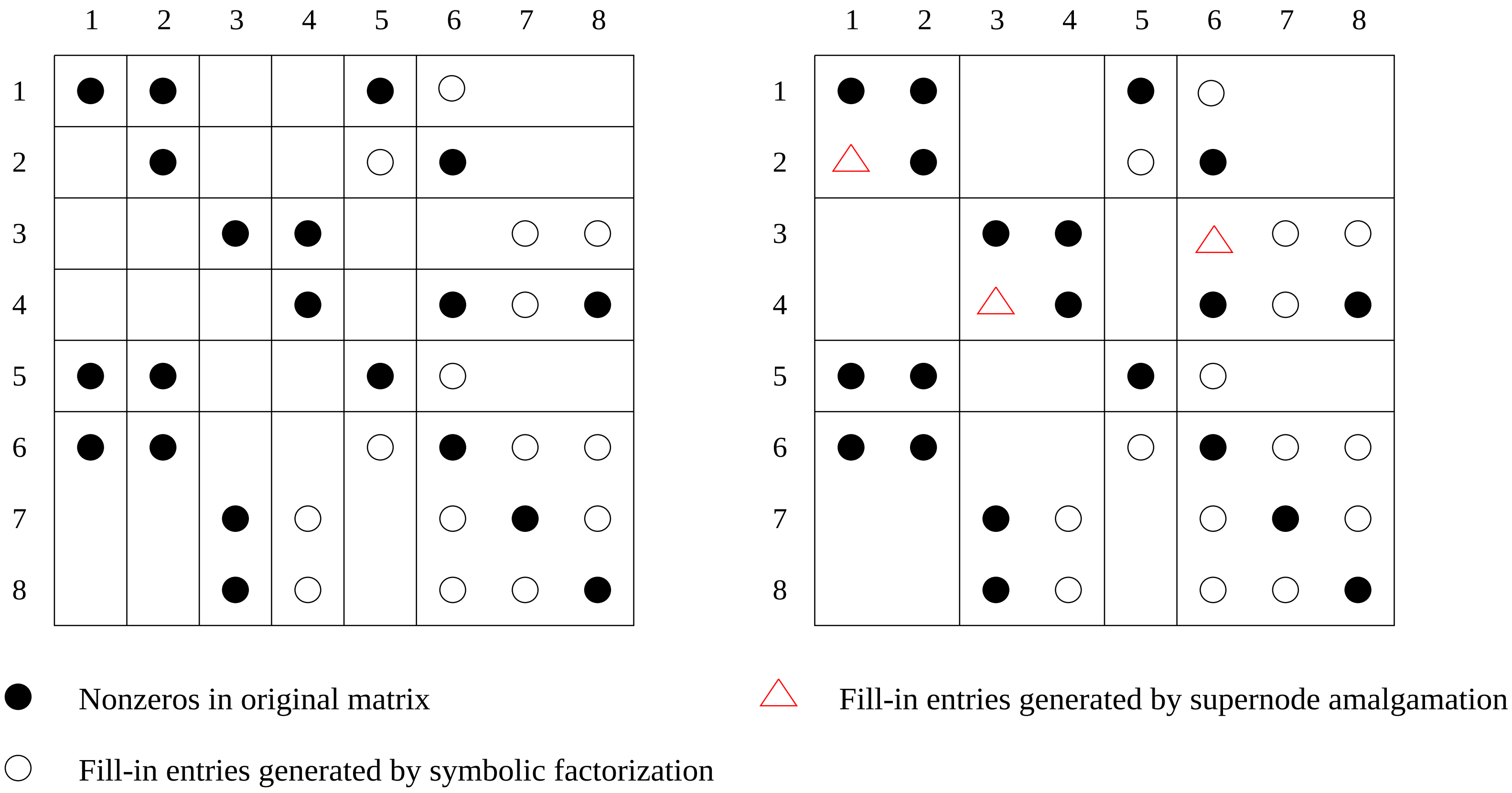
L/U supernode:



Algorithm: Group i and $i+1$ if i's parent is $i+1$ and $|L_i| = |L_{i+1}| + 1$.

Theorem: The produced supernode contains only dense subrows in the L part and dense subcolumns in the U part.

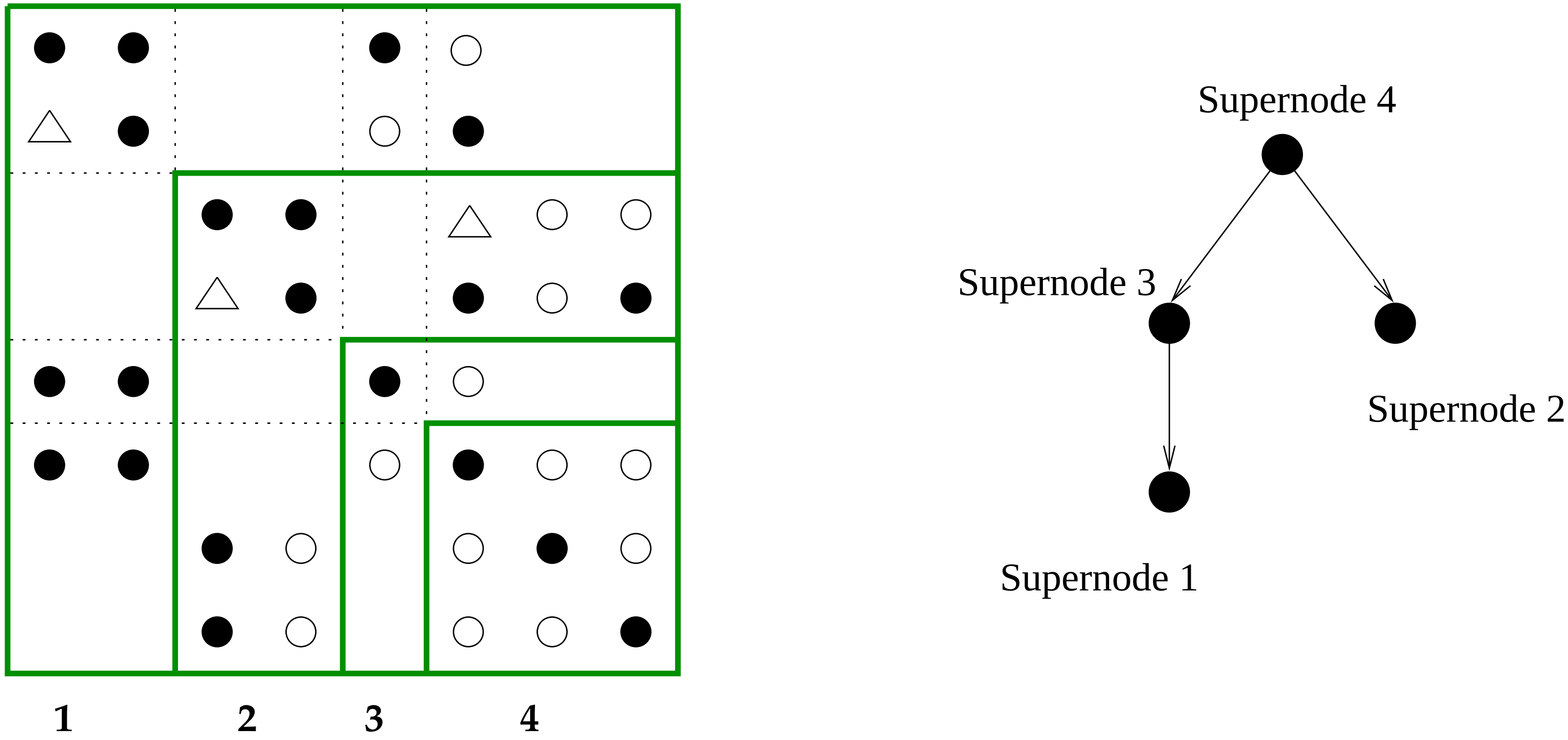
Use E-forest for supernode amalgamation



(a) (b)

Algorithm: Merge i and $i+1$ if $i+1$ is the parent of i and the number of extra fill-ins in L and U is within certain limit.

Supernodal E-forest for parallelism control



Theorem: Factor(k) can run concurrently with Factor(j) if they are not in the same path.

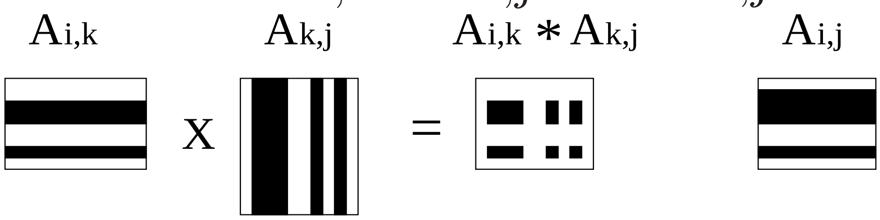
Key idea: Overlap updating stages and execute Factor() tasks as early as possible.

Supernodal GEMM

Dominating computation in *Update*() tasks:

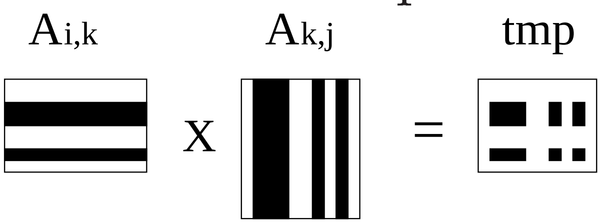
$$A_{i,j} = A_{i,j} - A_{i,k} * A_{k,j}$$

Difficulty for high performance: BLAS-3 GEMM is not directly applicable because of the structural difference between $A_{i,k} * A_{k,j}$ and $A_{i,j}$.

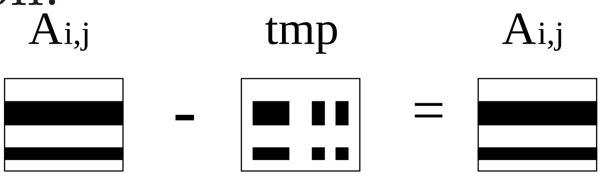


Supernodal GEMM:

1. Use BLAS-3 GEMM to compute $A_{i,k} * A_{k,j}$. Store the result in a temporal block.



2. Merge the temporal block with $A_{i,j}$ using subtraction.



Lazy space allocation

Observations:

1. For certain matrices, 10-24% nonzero blocks predicted by symbolic factorization remain zeros throughout numerical factorization.

2. Some nonzero blocks may become zero blocks due to pivoting.

$\Longrightarrow$ Lazy space allocation:

1. Delayed space allocation: acquire memory only when a block becomes truly nonzero during factorization.

2. Space reclamation: release space when nonzero blocks become zero.

Benefit of lazy space allocation

Illustration of Update(k,j) following symbolic pattern:

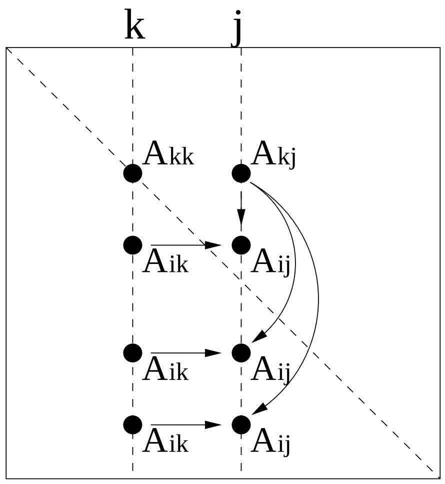
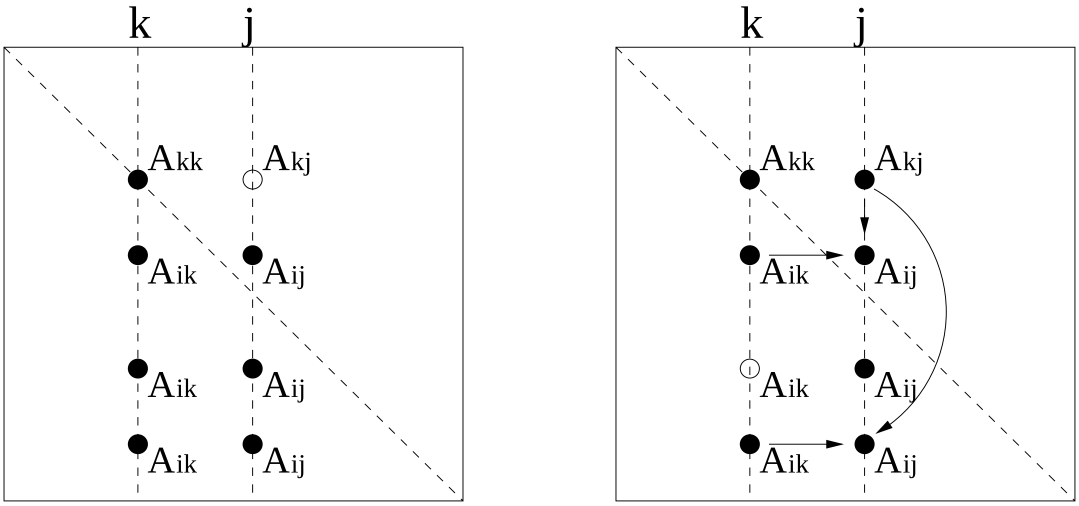


Illustration of Update(k,j) with lazy allocation:



(a) (b)

- ● nonzero blocks in numeric factorization
- ○ nonzero blocks recognized by symbolic factotization, but are actually zero blocks in numeric factorization

Test matrices and their statistics

Matrix	Order	$\|A\|$	factor entries/$\|A\|$ dynamic	static	$A^T * A$	Application domain
sherman5	3312	20793	12.03	15.70	20.42	Oil reservoir modeling
sherman3	5005	20033	22.13	31.20	39.24	Oil reservoir modeling
orsreg1	2205	14133	29.34	41.44	52.19	Oil reservoir simulation
saylr4	3564	22316	30.01	44.19	56.40	Oil reservoir modeling
goodwin	7320	324772	9.63	10.80	16.00	Fluid mechanics
e40r0100	17281	553562	14.76	17.32	26.48	Fluid dynamics
raefsky4	19779	1316789	20.36	28.06	35.68	Container modeling
af23560	23560	460598	30.39	44.39	57.40	Navier-Stokes Solver
fidap011	16614	1091362	23.36	24.55	31.21	Finite element modeling
vavasis3	41092	1683902	29.21	32.03	38.75	PDE
TIa	3432	25220	24.45	42.49	307.1	circuit simulation
TId	6136	53329	27.53	61.41	614.2	circuit simulation
TIb	18510	145149	91.84	278.34	1270.7	circuit simulation
memplus	17758	99147	71.26	168.77	215.19	circuit simulation
wang3	26064	177168	197.30	298.12	372.71	device simulation

Ordering: MMD on $A^T \times A$

Sequential performance on Sun UltraSparc1

Matrix	Space Requirement Ratio		Exec. Time Ratio	
	$\frac{LazyS^+}{SuperLU}$	$\frac{LazyS^+}{S^+}$	$\frac{LazyS^+}{SuperLU}$	$\frac{LazyS^+}{S^+}$
sherman5	0.90	0.97	0.853	0.959
sherman3	1.02	0.97	1.023	0.944
orsreg1	1.04	0.93	0.920	0.823
saylr4	1.09	0.94	0.964	0.870
goodwin	0.82	0.99	0.657	0.993
e40r0100	0.84	0.99	-	-
raefsky4	1.05	0.94	0.707	0.921
af23560	1.11	0.96	0.869	0.984
fidap011	0.81	0.99	-	-
TIa	1.17	0.86	0.675	0.629
TId	1.05	0.66	0.366	0.366
TIb	0.49	0.32	-	-
memplus	0.91	0.50	-	-
wang3	-	0.81	-	-

Performance without lazy allocation

MFLOPS performance on 300Mhz Cray T3E:

Matrix	P=8	P=16	P=32	P=64	P=128
goodwin	403.5	603.4	736.0	797.3	826.8
e40r0100	443.2	727.8	992.8	1204.8	1272.8
raefsky4	568.2	1072.5	1930.3	3398.1	5133.6
af23560	432.1	753.2	1161.3	1518.9	1844.7
vavasis3	937.3	1823.7	3230.8	5516.2	8256.0
TIa	222.2	360.6	490.1	579.2	616.5
TId	188.1	329.2	484.5	658.4	778.1
TIb	65.25	127.9	230.8	431.4	696.0
memplus	429.0	834.6	1495.4	2602.3	3882.2
wang3	498.0	973.3	1783.8	3284.9	5319.4

Performance impact of lazy allocation

Matrix	P=8	P=16	P=32	P=64	P=128
goodwin	-7.28%	-8.29%	-8.10%	-1.17%	-4.69%
e40r0100	-6.81%	-8.81%	-11.34%	-8.84%	-7.13%
raefsky4	3.41%	2.52%	-0.42%	-1.82%	-5.02%
af23560	-3.17%	-3.98%	-9.72%	-4.56%	-13.76%
vavasis3	7.65%	-1.79%	5.02%	-2.16%	-6.13%
TIa	13.16%	10.42%	2.63%	-2.94%	-6.06%
TId	35.15%	23.81%	9.28%	-2.50%	-9.59%
TIb	352.20%	270.26%	209.27%	133.69%	78.10%
memplus	136.43%	115.38%	87.09%	61.84%	35.24%
wang3	10.51%	7.57%	3.52%	1.53%	-5.17%

$+$: improvement; $-$: degradation

MFLOPS performance on 450MHz T3E

Matrix	8 nodes	128 nodes
vavasis3	1492.9	10004.0
TIb	555.7	2515.7
memplus	1439.4	6548.4
wang3	757.8	6261.0

Megaflop count does not include extra operation introduced by static symbolic factorization and amalgamation.

Conclusions

Contributions:

- Fastest parallel sparse *LU* factorization on distributed memory machines with reasonable space cost. Performance record: up to 10Gflops on 128 nodes 450MHz Cray T3E.
- E-forest guided matrix partitioning and parallel execution.
- Supernodal GEMM for improving kernel computation speed.
- Lazy space allocation reduces memory requirements and unnecessary computation when fill-in estimation is excessive.

More information:

- http://www.cs.ucsb.edu/research/S+