Sparse linear solvers

Laura Grigori

ALPINES
INRIA and LJLL, UPMC
On sabbatical at UC Berkeley

March 2015

Plan

Sparse linear solvers
- Sparse matrices and graphs
- Classes of linear solvers

Sparse Cholesky factorization for SPD matrices
- Combinatorial tools: undirected graphs, elimination trees
- Parallel Cholesky factorization
- Lower bounds for sparse Cholesky factorization

Extra slides: Sparse LU factorization
- Combinatorial tools: directed and bipartite graphs
- LU factorization on parallel machines

Plan

Sparse matrices and graphs

- Most matrices arising from real applications are sparse.
- A 1M-by-1M submatrix of the web connectivity graph, constructed from an archive at the Stanford WebBase.

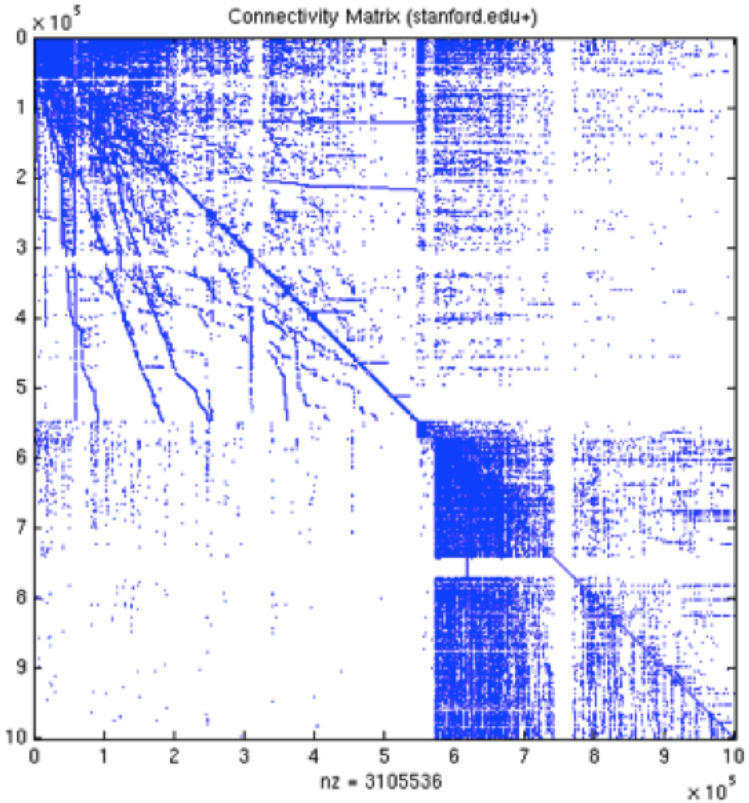


Figure : Nonzero structure of the matrix

Sparse matrices and graphs

- Most matrices arising from real applications are sparse.
- GHS class: Car surface mesh, $n = 100196, nnz(A) = 544688$

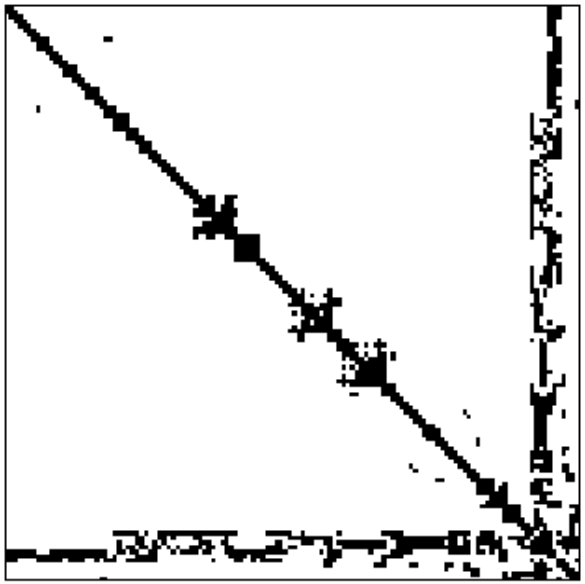


Figure : Nonzero structure of the matrix

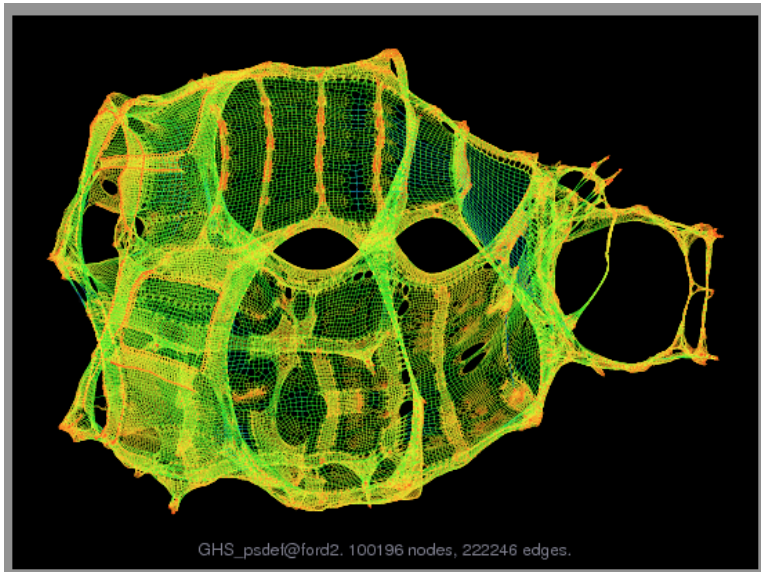


Figure : Its undirected graph

Examples from Tim Davis's Sparse Matrix Collection,
`http://www.cise.ufl.edu/research/sparse/matrices/`

Sparse matrices and graphs

- Semiconductor simulation matrix from Steve Hamm, Motorola, Inc. circuit with no parasitics, $n = 105676$, $nnz(A) = 513072$

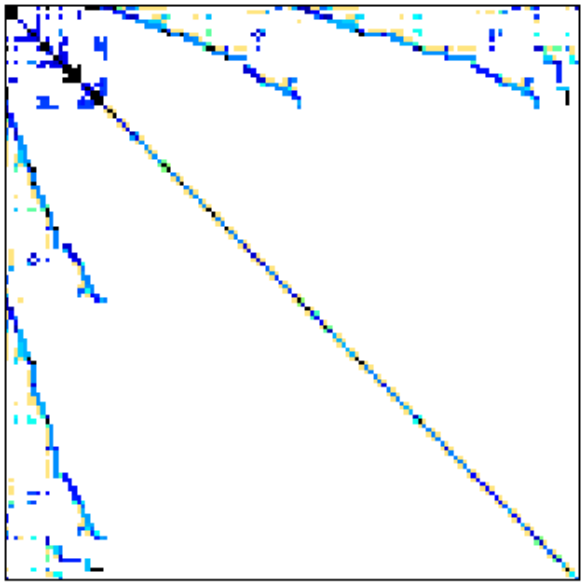


Figure : Nonzero structure of the matrix

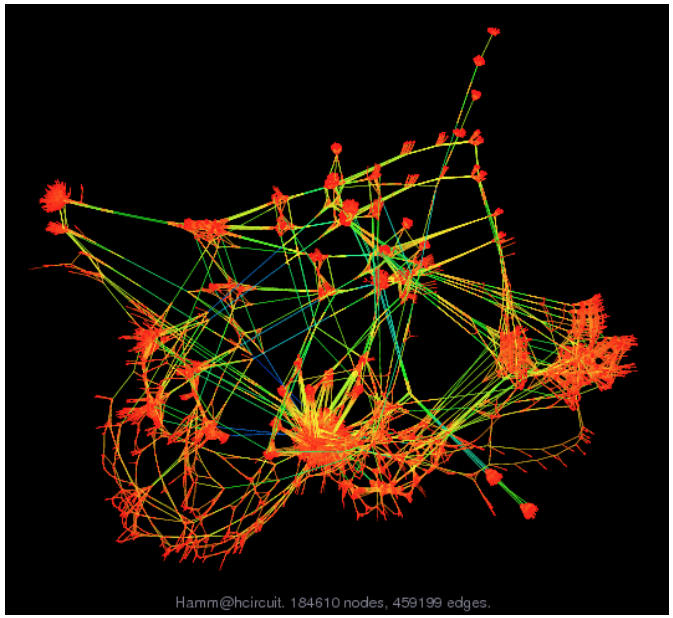


Figure : Its undirected graph

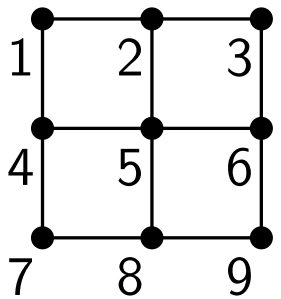
Examples from Tim Davis's Sparse Matrix Collection,
`http://www.cise.ufl.edu/research/sparse/matrices/`

Symmetric sparse matrices and graphs

- The structure of a square symmetric matrix A with nonzero diagonal can be represented by an undirected graph $G(A) = (V, E)$ with
  - n vertices, one for each row/column of A
  - an edge (i, j) for each nonzero $a_{ij}, i > j$

$A =$

	1	2	3	4	5	6	7	8	9
1	x	x		x					
2	x	x	x		x				
3		x	x			x			
4	x			x	x		x		
5		x		x	x	x		x	
6			x		x	x			x
7				x			x	x	
8					x		x	x	x
9						x		x	x

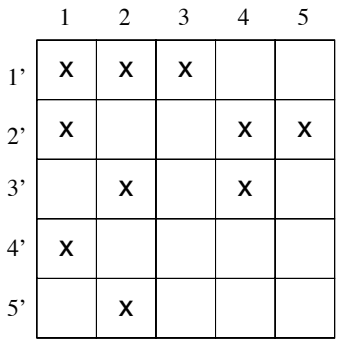


$G(A)$

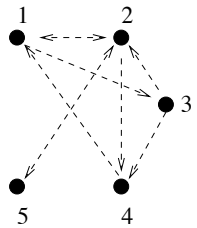
Notation: upper case (A) - matrices; lower case (a_{ij}) - elements

Nonsymmetric sparse matrices and graphs

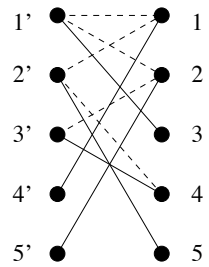
- The structure of a nonsymmetric matrix A of size $n \times n$ can be represented by
  - a directed graph $G(A) = (V, E)$ with
    - n vertices, one for each column of A
    - an edge from i to j for each nonzero a_{ij}
  - a bipartite graph $H(A) = (V, E)$ with
    - $2n$ vertices, for n rows and n columns of A
    - an edge (i', j) for each nonzero a_{ij}
  - a hypergraph (not described further in this class)



A



G(A)



H(A)

Computing with sparse matrices - guiding principles

- Store nonzero elements

$$A = \begin{pmatrix} 1.1 & & 1.3 & & \\ & 2.2 & & 2.4 & \\ & 3.2 & & & 3.5 \\ 4.1 & & & 4.4 & \\ 5.1 & & 5.3 & & 5.5 \end{pmatrix}$$

  - Compressed sparse formats by columns (CSC), rows (CSR), or coordinates
  - For A of size $n \times n$, CSC uses one aray of size $n+1$ and two arrays of size $nnz(A)$:

$$\begin{array}{rc} \mathit{ColPtr} & \begin{pmatrix} 1 & 4 & 6 & 8 & 10 & 12 \end{pmatrix} \\ \mathit{RowInd} & \left(\begin{array}{ccc|cc|cc|cc|cc} 1 & 4 & 5 & 2 & 3 & 1 & 5 & 2 & 4 & 3 & 5 \\ \mathit{Vals} & 1.1 & 4.1 & 5.1 & 2.2 & 3.2 & 1.3 & 5.3 & 2.4 & 4.4 & 3.5 & 5.5 \end{array}\right) \end{array}$$

- Compute flops only on nonzeros elements
- Identify and exploit parallelism due to the sparsity of the matrix

Sparse linear solvers

Direct methods of factorization

- For solving $Ax = b$, least squares problems
  - Cholesky, LU, QR, LDL^T factorizations
- Limited by fill-in/memory consumption and scalability

Iterative solvers

- For solving $Ax = b$, least squares, $Ax = \lambda x$, SVD
- When only multiplying A by a vector is possible
- Limited by accuracy/convergence

Hybrid methods

As domain decomposition methods

Examples of direct solvers

A non-complete list of solvers and their characteristics:

- PSPASES: for SPD matrices, distributed memory.
  `http://www-users.cs.umn.edu/~mjoshi/pspases/`
- UMFPACK / SuiteSparse (Matlab, Google Ceres) - symmetric/unsymmetric, LU, QR, multicores/GPUs.
  `http://faculty.cse.tamu.edu/davis/suitesparse.html`
- SuperLU: unsymmetric matrices, shared/distributed memory.
  `http://crd-legacy.lbl.gov/~xiaoye/SuperLU/`
- MUMPS: symmetric/unsymmetric, distributed memory.
  `http://mumps.enseeiht.fr/`
- Pardiso (Intel MKL): symmetric/unsymmetric, shared/distributed memory.
  `http://www.pardiso-project.org/`

For a survey, see `http://crd.lbl.gov/~xiaoye/SuperLU/SparseDirectSurvey.pdf`.

Plan

Sparse linear solvers

Sparse Cholesky factorization for SPD matrices

- Combinatorial tools: undirected graphs, elimination trees
- Parallel Cholesky factorization
- Lower bounds for sparse Cholesky factorization

Extra slides: Sparse LU factorization

To get started: algebra of LU factorization

LU factorization

Compute the factorization PA = LU

Example

Given the matrix

$$A = \begin{pmatrix} 3 & 0 & 3 \\ 6 & 7 & 0 \\ 9 & 12 & 3 \end{pmatrix}$$

The first step of the LU factorization is performed as

$$M_1 = \begin{pmatrix} 1 & & \\ -2 & 1 & \\ -3 & & 1 \end{pmatrix}, \quad M_1 A = \begin{pmatrix} 3 & 0 & 3 \\ 0 & 7 & -6 \\ 0 & 12 & -6 \end{pmatrix}$$

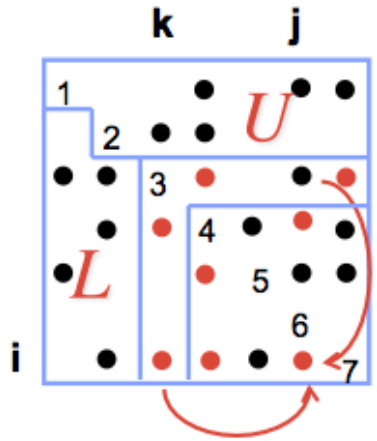
Fill-in elements

Are elements which are zero in A, but become nonzero in L or U (as -6 above).

Sparse LU factorization

Right looking factorization of A by rows

for $k = 1 : n - 1$ **do**
  Permute row i and row k, where a_{ik} is element of maximum magnitude in $A(k : n, k)$
  for $i = k + 1 : n$ st $a_{ik} \neq 0$ **do**
    /* store in place l_{ik} */
    $a_{ik} = a_{ik}/a_{kk}$
    /* add a multiple of row k to row i */
    for $j = k + 1 : n$ st $a_{kj} \neq 0$ **do**
      $a_{ij} = a_{ij} - a_{ik} * a_{kj}$
    end for
  end for
end for



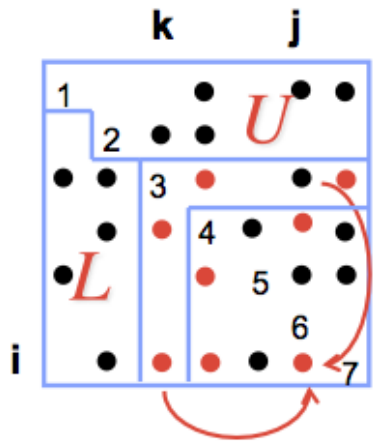
Observations

- The order of the indices i, j, k can be changed, leading to different algorithms:
  - computing the factorization by rows, by columns, or by sub-matrices,
  - using a left looking, right looking, or multifrontal approach.

Sparse LU factorization with partial pivoting

Factorization by columns

for $k = 1 : n-1$ **do**
  Permute row i and row k, where a_{ik} is element of maximum magnitude in $A(k:n, k)$
  for $i = k+1 : n$ st $a_{ik} \neq 0$ **do**
    $a_{ik} = a_{ik}/a_{kk}$
  end for
  for $j = k+1 : n$ st $a_{kj} \neq 0$ **do**
    for $i = k+1 : n$ st $a_{ik} \neq 0$ **do**
      $a_{ij} = a_{ij} - a_{ik}a_{kj}$
    end for
  end for
end for



A simpler case first: SPD matrices

A is symmetric and positive definite (SPD) if

- $A = A^T$,
- all its eigenvalues are positive,
- or equivalently, A has a Cholesky factorization, $A = LL^T$.

Some properties of an SPD matrix A

- There is no need to pivot for accuracy (just performance) during the Cholesky factorization.
- For any permutation matrix P, PAP^T is also SPD.

Sparse Cholesky factorization

- The algebra can be written as:

$$A = \begin{pmatrix} a_{11} & A_{21}^T \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} \sqrt{a_{11}} & \\ A_{21}./\sqrt{a_{11}} & L_{22} \end{pmatrix} \cdot \begin{pmatrix} \sqrt{a_{11}} & A_{21}^T./\sqrt{a_{11}} \\ & L_{22}^T \end{pmatrix}$$

- Compute and store only the lower triangular part since $U = L^T$.

Algorithm

for $k = 1 : n-1$ **do**
  $a_{kk} = \sqrt{a_{kk}}$
  /* $factor(k)$ */
  for $i = k+1 : n$ st $a_{ik} \neq 0$ **do**
    $a_{ik} = a_{ik}/a_{kk}$
  end for
  for $i = k+1 : n$ st $a_{ik} \neq 0$ **do**
    $update(k, i)$
    for $j = i : n$ st $a_{kj} \neq 0$ **do**
      $a_{ij} = a_{ij} - a_{ik}a_{jk}$
    end for
  end for

end for

Filled graph $G^+(A)$

- Given $G(A) = (V, E)$, $G^+(A) = (V, E^+)$ is defined as:
  there is an edge $(i, j) \in G^+(A)$ iff there is a path from i to j in $G(A)$ going through lower numbered vertices.
- Definition holds also for directed graphs (LU factorization).

- $G(L + L^T) = G^+(A)$, ignoring cancellations.
- $G^+(A)$ is chordal (every cycle of length at least four has a chord, an edge connecting two non-neighboring nodes).
- Conversely, if $G(A)$ is chordal, then there is a perfect elimination order, that is a permutation P such that $G(PAP^T) = G^+(PAP^T)$.

- References: [Parter, 1961, Rose, 1970, Rose and Tarjan, 1978]

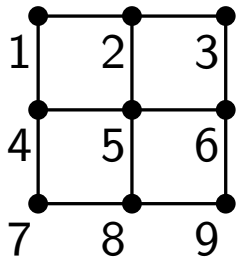
Filled graph $G^+(A)$

$A =$

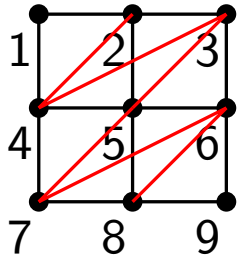
	1	2	3	4	5	6	7	8	9
1	x	x		x					
2	x	x	x		x				
3		x	x			x			
4	x			x	x		x		
5		x		x	x	x		x	
6			x		x	x			x
7				x			x	x	
8					x		x	x	x
9						x		x	x

$L + L^T =$

	1	2	3	4	5	6	7	8	9
1	x	x		x					
2	x	x	x	x	x				
3		x	x	x	x	x			
4	x	x	x	x	x		x		
5		x	x	x	x	x	x	x	
6			x		x	x	x	x	x
7				x	x	x	x	x	
8					x	x	x	x	x
9						x		x	x



$G(A)$



$G^+(A)$

Steps of sparse Cholesky factorization

1. Order rows and columns of A to reduce fill-in

2. Symbolic factorization: based on eliminaton trees
   - Compute the elimination tree (in nearly linear time in $nnz(A)$)
   - Allocate data structure for L
   - Compute the nonzero structure of the factor L, in $O(nnz(L)$

3. Numeric factorization
   - Exploit memory hierarchy
   - Exploit parallelism due to sparsity

4. Triangular solve

Order columns/rows of A

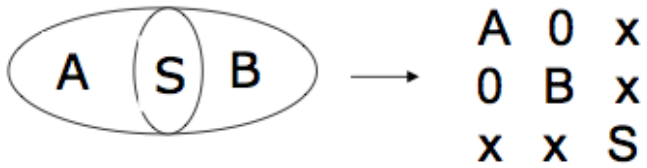
Strategies applied to the graph of A for Cholesky,
to the graph of $A^T A$ for LU with partial pivoting.

Local strategy: minimum degree [Tinney/Walker '67]

- Minimize locally the fill-in.
- Choose at each step (for 1 to n) the node of minimum degree.

Global strategy: graph partitioning approach

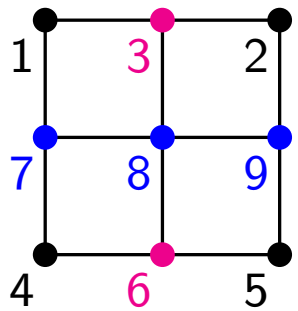
- Nested dissection [George, 1973]
  - First level: find the smallest possible separator S, order last
  - Recurse on A and B
- Multilevel schemes [Barnard/Simon '93, Hendrickson/Leland '95, Karypis/Kumar '95].



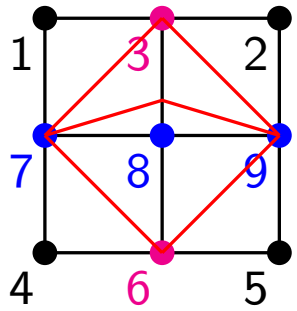
Nested dissection on our 9×9 structured matrix

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \left(\begin{array}{ccc|ccc|ccc} x & & x & & & & x & & \\ & x & x & & & & & & x \\ \hline x & x & x & & & & & x & \\ \hline & & & x & & x & x & & \\ & & & & x & x & & & x \\ \hline & & & x & x & x & & x & \\ \hline x & & & x & & & x & x & \\ & & x & & & x & x & x & x \\ & x & & & x & & & x & x \end{array} \right) \end{array},$$

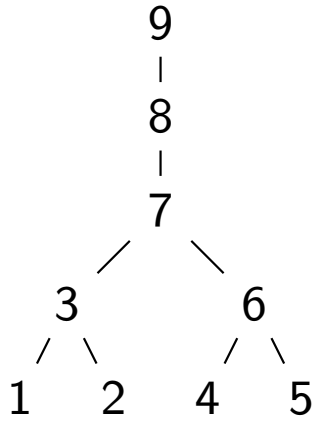
$$L + L^T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \left(\begin{array}{ccc|ccc|ccc} x & & x & & & & x & & \\ & x & x & & & & & & x \\ \hline x & x & x & & & & x & x & x \\ \hline & & & x & & x & x & & \\ & & & & x & x & & & x \\ \hline & & & x & x & x & x & x & x \\ \hline x & & x & x & & x & x & x & x \\ & & x & & & x & x & x & x \\ & x & x & & x & x & x & x & x \end{array} \right) \end{array}$$



$G(A)$



$G^+(A)$



$T(A)$

Elimination tree (etree)

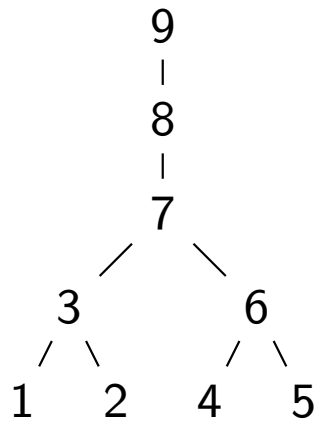
Definition ([Schreiber, 1982] and also [Duff, 1982])

Given $A = LL^T$, the etree $T(A)$ has the same node set as $G(A)$, and k is the parent of j iff

$$k = \min\{i > j : l_{ij} \neq 0\}$$

$L + L^T =$

	1	2	3	4	5	6	7	8	9
1	x		x				x		
2		x	x						x
3	x	x	x				x	x	x
4				x		x	x		
5					x	x			x
6				x	x	x	x	x	x
7	x		x	x		x	x	x	x
8			x			x	x	x	x
9		x	x		x	x	x	x	x



$T(A)$

Elimination tree (etree)

Definition ([Schreiber, 1982] and also [Duff, 1982])

Given $A = LL^T$, the etree $T(A)$ has the same node set as $G(A)$, and k is the parent of j iff

$$k = \min\{i > j : l_{ij} \neq 0\}$$

Properties (ignoring cancellations), for more details see e.g. [Liu, 1990]

- $T(A)$ is a spanning tree of the filled graph $G^+(A)$.
- $T(A)$ is the result of the transitive reduction of the directed graph $G(L^T)$.
- $T(A)$ of a connected graph $G(A)$ is a depth first search tree of $G^+(A)$ (with specific tie-breaking rules).

Elimination tree (contd)

Complexity

- Can be computed in $O(nnz(A)\alpha(nnz(A), n))$, where $\alpha()$ is the inverse of Ackerman's function.

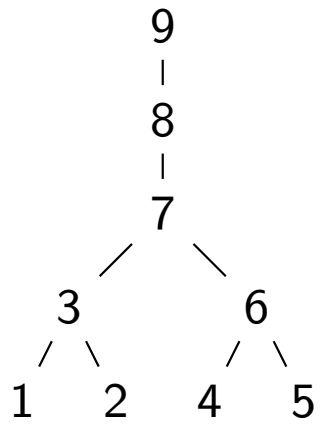
- Can be used to
  - compute # nonzeros of each column/row of L (same complexity),
  - identify columns with similar structure (supernodes), (same complexity)
  - compute nonzero structure of L, in $O(nnz(L))$

Column dependencies and the elimination tree

- If $l_{jk} \neq 0$, then
  - $Factor(k)$ needs to be computed before $Factor(j)$.
  - k is an ancestor of j in $T(A)$.
- Columns belonging to disjoint subtrees can be factored independently.
- Topological orderings of $T(A)$ (that number children before their parent)
  - preserve the amount of fill, the flops of the factorization, the structure of $T(A)$

$L + L^T =$

	1	2	3	4	5	6	7	8	9
1	x		x				x		
2		x	x						x
3	x	x	x				x	x	x
4				x		x	x		
5					x	x			x
6				x	x	x	x	x	x
7	x		x	x		x	x	x	x
8			x			x	x	x	x
9		x	x		x	x	x	x	x

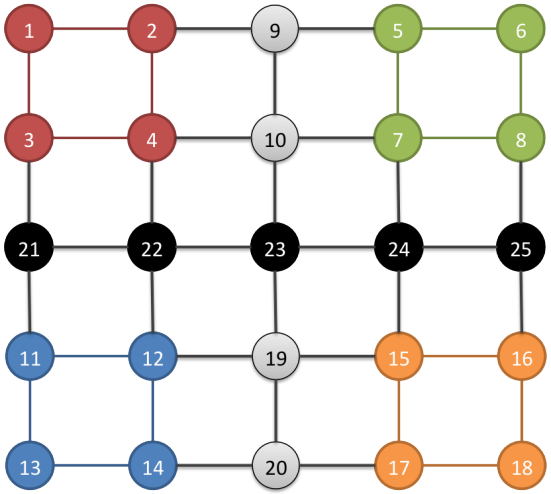


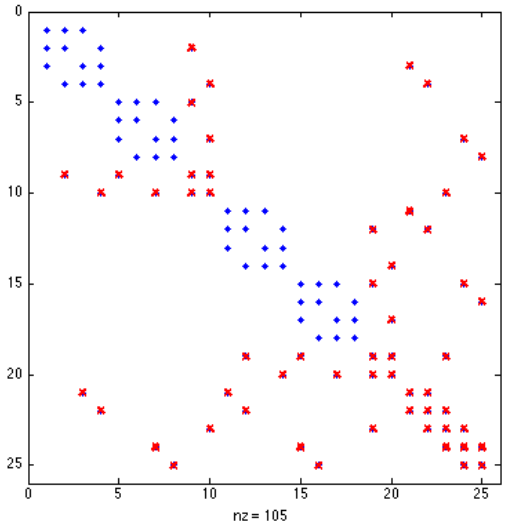
$T(A)$

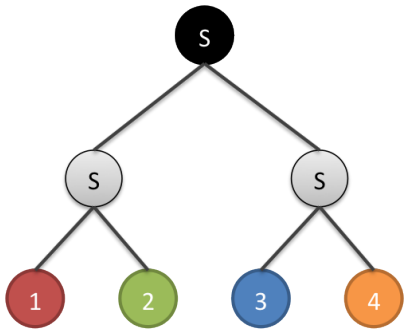
Nested dissection and separator tree

Separator tree:

- Combines together nodes belonging to a same separator, or to a same disjoint graph







Some available packages (see also lecture 14 on graph partitioning):

- Metis, Parmetis (`http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`)
- Scotch, Ptscotch (`www.labri.fr/perso/pelegrin/scotch/`)

Numeric factorization - multifrontal approach

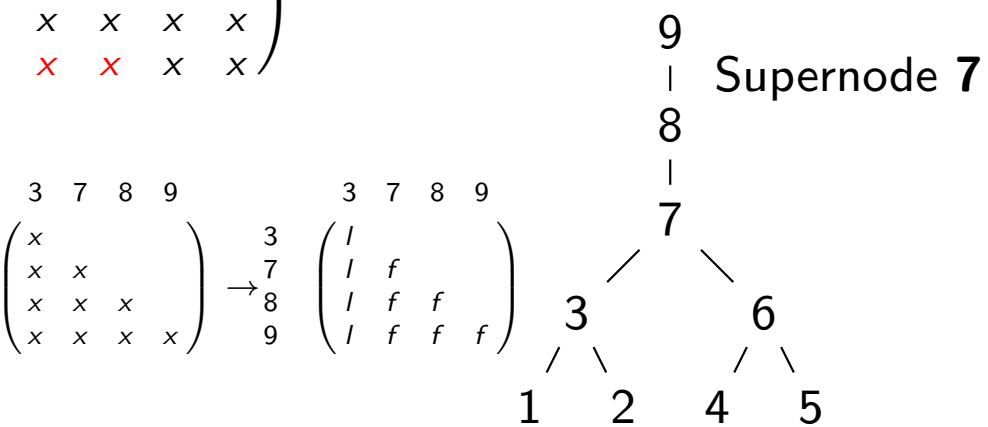
- Driven by the separator tree of A, a supernodal elimination tree.
- The Cholesky factorization is performed during a postorder traversal of the separator tree.
- At each node k of the separator tree:
  - A frontal matrix F_k is formed by rows and columns involved at step k of factorization:
    - rows that have their first nonzero in column k of A,
    - contribution blocks (part of frontal matrices) from children in $T(A)$.
  - The new frontal matrix is obtained by an extend-add operation.
  - The first rows/columns of F_k corresponding to supernode k are factored.

Numeric factorization - an example

$$L + L^T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} x & & x & & & & x & & \\ & x & x & & & & & & x \\ x & x & x & & & & x & x & x \\ & & & x & & x & x & & \\ & & & & x & x & & & x \\ & & & x & x & x & x & x & x \\ x & & x & x & & x & x & x & x \\ & & x & & & x & x & x & x \\ & x & x & & x & x & x & x & x \end{pmatrix} \end{array}$$

$$F_{\{7,8,9\}} = \begin{array}{c} \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccc} 7 & 8 & 9 \end{array} \\ \begin{pmatrix} x & & \\ x & x & \\ x & x & x \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccc} 7 & 8 & 9 \end{array} \\ \begin{pmatrix} l & & \\ l & l & \\ l & l & l \end{pmatrix} \end{array}$$

$$F_3 = \begin{array}{c} \\ 3 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{cccc} 3 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} x & & & \\ x & x & & \\ x & x & x & \\ x & x & x & x \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \\ 3 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{cccc} 3 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} l & & & \\ l & f & & \\ l & f & f & \\ l & f & f & f \end{pmatrix} \end{array}$$



$$F_1 = \begin{array}{c} \\ 1 \\ 3 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 3 & 7 \end{array} \\ \begin{pmatrix} x & & \\ x & x & \\ x & x & x \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \\ 1 \\ 3 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 3 & 7 \end{array} \\ \begin{pmatrix} l & & \\ l & f & \\ l & f & f \end{pmatrix} \end{array} \quad F_2 = \begin{array}{c} \\ 2 \\ 3 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccc} 2 & 3 & 9 \end{array} \\ \begin{pmatrix} x & & \\ x & x & \\ x & x & x \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \\ 2 \\ 3 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccc} 2 & 3 & 9 \end{array} \\ \begin{pmatrix} l & & \\ l & f & \\ l & f & f \end{pmatrix} \end{array}$$

Notation used for frontal matrices F_k:

- x - elements obtained by the extend-add operation,
- l - elements of L computed at node k, f - elements of frontal matrix that will be passed to parent of node k.

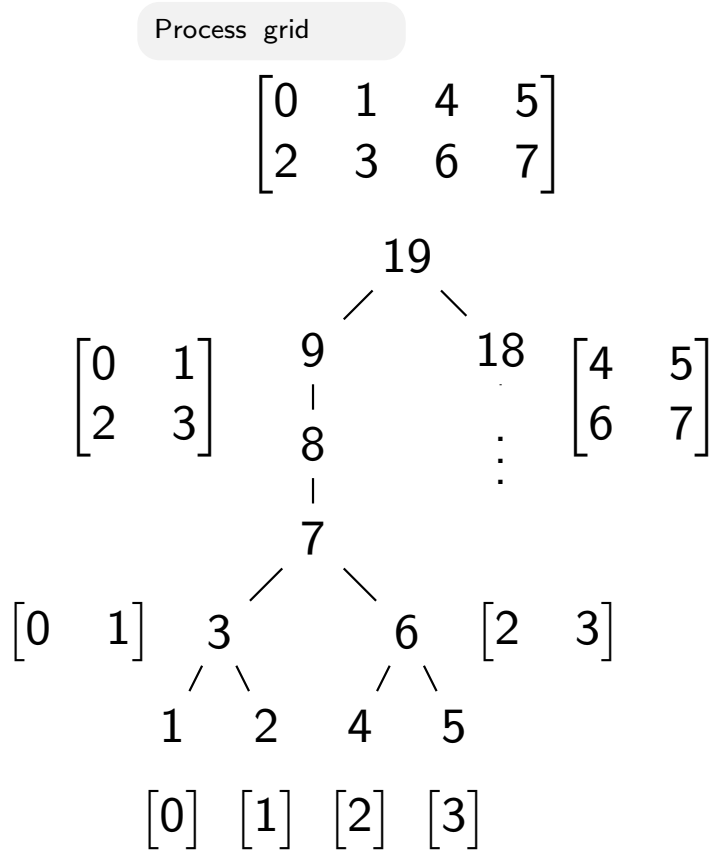
Numeric factorization - PSPASES [Gupta et al., 1995]

- Based on subtree to subcube mapping [George et al., 1989] applied on the separator tree

Subtree to subcube mapping

1. Assign all the processors to the root.
2. Assign to each subtree half of the processors.
3. Go to Step 1 for each subtree which is assigned more than one processor.

The figure displays the process grid used by PSPASES.



Numeric factorization - PSPASES [Gupta et al., 1995]

- Subtree to subcube mapping and bitmask based cyclic distribution:

  Starting at the last level of the separator tree (bottom up traversal), let $i = 1$

  for each two consecutive levels $k, k-1$, based on value of i-th LSB of column/row indices

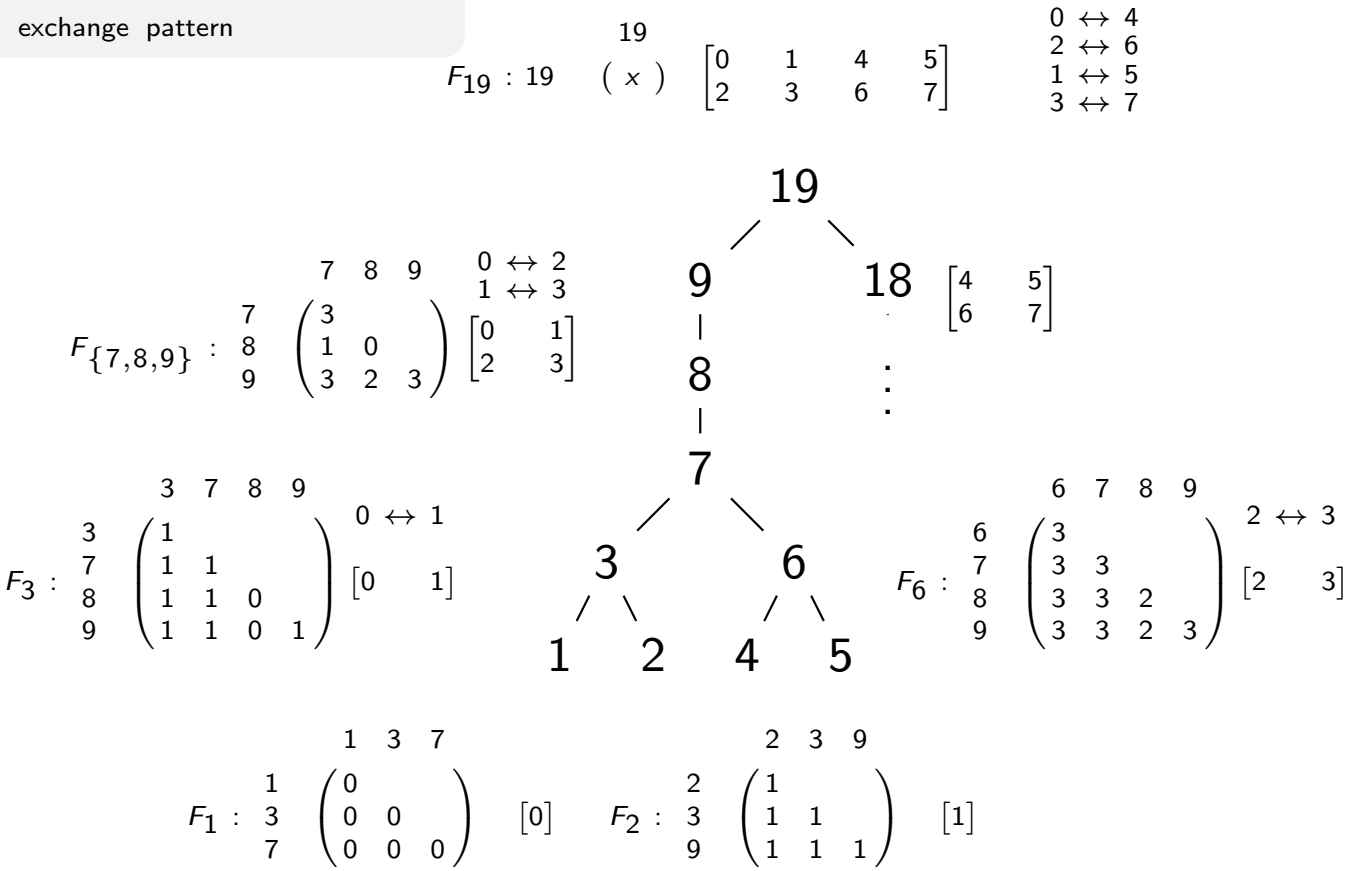
  - For level k:
    Map all even columns to subcube with lower processor numbers
    Map all odd columns to subcube with higher processor numbers
  - For level $k-1$:
    Map all even rows to subcube with lower processor numbers
    Map all odd rows to subcube with higher processor numbers
  - Let $i = i + 1$

PSPASES uses a bitmask based block-cyclic distribution.

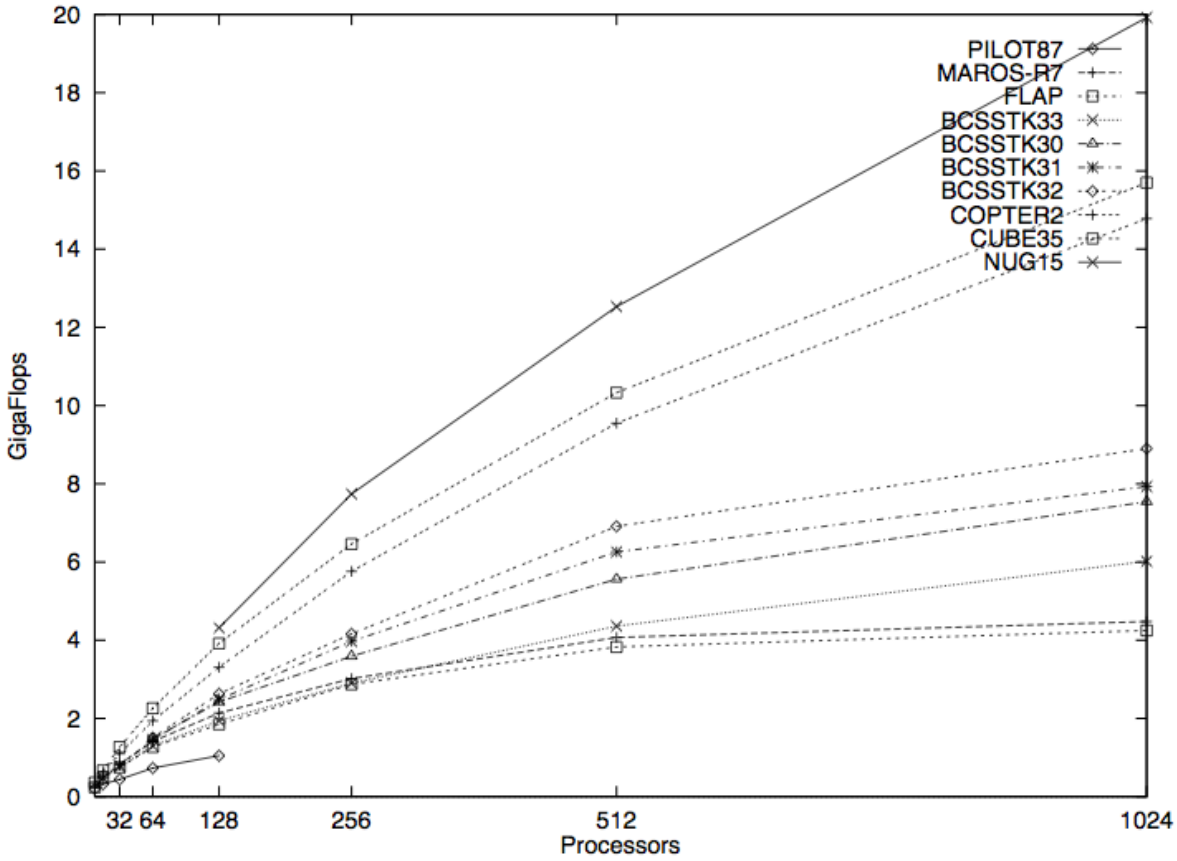
Numeric factorization - PSPASES [Gupta et al., 1995]

- Based on subtree to subcube mapping [George et al., 1989].
- Extend-add operation requires each processor to exchange half of its data with a corresponding processor from the other half of the grid.

Data distribution, process grid and data exchange pattern



Performance results on Cray T3D



Results from [Gupta et al., 1995]

Performance - break-down of the various phases

		Distributed Computation		
	p	Local Comp.	Factorization	Extend-Add
BCSSTK31	64	0.17	1.34	0.58
	128	0.06	0.90	0.32
	256	0.02	0.61	0.18
CUBE35	64	0.15	3.74	0.71
	128	0.06	2.25	0.43
	256	0.01	1.44	0.24

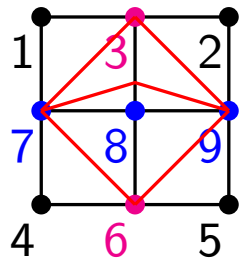
Lower bounds on communication for sparse LA

- More difficult than the dense case
  - For example computing the product of two (block) diagonal matrices involves no communication in parallel
- Lower bound on communication from dense linear algebra is loose
- Very few existing results:
  - Lower bounds for parallel multiplication of sparse random matrices [Ballard et al., 2013]
  - Lower bounds for Cholesky factorization of model problems [Grigori et al., 2010]

Lower bounds on communication for Cholesky

- Consider A of size $k^s \times k^s$ results from a finite difference operator on a regular grid of dimension $s \geq 2$ with k^s nodes.
- Its Cholesky L factor contains a dense lower triangular matrix of size $k^{s-1} \times k^{s-1}$.

$$L + L^T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} x & & x & & & & x & & \\ & x & x & & & & & & x \\ x & x & x & & & & x & x & x \\ & & & x & & x & x & & \\ & & & & x & x & & & x \\ & & & x & x & x & x & x & x \\ x & & x & x & & x & x & x & x \\ & & x & & & x & x & x & x \\ & x & x & & x & x & x & x & x \end{pmatrix} \end{array}$$



$G^+(A)$

- Computing the Cholesky factorization of the $k^{s-1} \times k^{s-1}$ matrix dominates the computation.

Lower bounds on communication

- This result applies more generally to matrix A whose graph $G = (V, E)$, $|V| = n$ has the following property for some l:
  - if every set of vertices $W \subset V$ with $n/3 \leq |W| \leq 2n/3$ is adjacent to at least l vertices in $V - W$,
  - then the Cholesky factor of A contains a dense $l \times l$ submatrix.

Lower bounds on communication

For the Cholesky factorization of a $k^s \times k^s$ matrix resulting from a finite difference operator on a regular grid of dimension $s \geq 2$ with k^s nodes:

$$\#words \geq \Omega\left(\frac{W}{\sqrt{M}}\right), \qquad \#messages \geq \Omega\left(\frac{W}{M^{3/2}}\right)$$

- Sequential algorithm
  - $W = k^{3(s-1)}/3$ and M is the fast memory size
- Work balanced parallel algorithm executed on P processors
  - $W = \frac{k^{3(s-1)}}{3P}$ and $M \approx nnz(L)/P$

Why / how PSPASES attains optimality

- For each node in the separator tree, the communication in the Cholesky factorization dominates the communication in the extend-add step.
- Optimal dense Cholesky factorization needs to be used for each multifrontal matrix ($n \times n$, P procs).
  - optimal block size - minimize communication while increasing flops by a lower order term

$$b = \frac{n}{\sqrt{P}} \log_2^{-2} \sqrt{P}$$

Optimal sparse Cholesky factorization

- Results for $n \times n$ matrix resulting from 2D and 3D regular grids.
- Analysis assumes local memory per processor is $M = O(n \log n/P)$- 2D case and $M = O(n^{4/3}/P)$- 3D case.

	PSPASES	PSPASES with optimal layout	Lower bound
2D grids			
# flops	$O\left(\frac{n^{3/2}}{P}\right)$	$O\left(\frac{n^{3/2}}{P}\right)$	$\Omega\left(\frac{n^{3/2}}{P}\right)$
# words	$O(\frac{n}{\sqrt{P}})$	$O\left(\frac{n}{\sqrt{P}} \log P\right)$	$\Omega\left(\frac{n}{\sqrt{P \log n}}\right)$
# messages	$O(\sqrt{n})$	$O\left(\sqrt{P} \log^3 P\right)$	$\Omega\left(\frac{\sqrt{P}}{(\log n)^{3/2}}\right)$
3D grids			
# flops	$O\left(\frac{n^2}{P}\right)$	$O\left(\frac{n^2}{P}\right)$	$\Omega\left(\frac{n^2}{P}\right)$
# words	$O(\frac{n^{4/3}}{\sqrt{P}})$	$O\left(\frac{n^{4/3}}{\sqrt{P}} \log P\right)$	$\Omega\left(\frac{n^{4/3}}{\sqrt{P}}\right)$
# messages	$O(n^{2/3})$	$O\left(\sqrt{P} \log^3 P\right)$	$\Omega\left(\sqrt{P}\right)$

Optimal sparse Cholesky factorization: summary

- PSPASES with an optimal layout attains the lower bound in parallel for 2D/3D regular grids:
  - Uses nested dissection to reorder the matrix
  - Distributes the matrix using the subtree to subcube algorithm
  - The factorization of every dense multifrontal matrix is performed using an optimal dense Cholesky factorization
- Sequential multifrontal algorithm attains the lower bound
  - The factorization of every dense multifrontal matrix is performed using an optimal dense Cholesky factorization

Conclusions

- Direct methods of factorization are very stable, but have limited scalability (up to hundreds/a few thousands of cores).

- Open problems:
  - Develop more scalable algorithms.
  - Identify lower bounds on communication for other operations: LU, QR, etc.

Plan

Sparse linear solvers

Sparse Cholesky factorization for SPD matrices

Extra slides: Sparse LU factorization

- Combinatorial tools: directed and bipartite graphs
- LU factorization on parallel machines

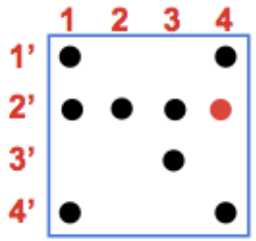
Structure prediction for $A = LU$

- A is square, unsymmetric, and has a nonzero diagonal.
- Nonzero structure of L and U can be determined prior to the numerical factorization from the structure of A [Rose and Tarjan, 1978].

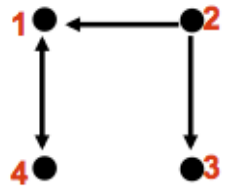
Filled graph $G^+(A)$:

- edges from rows to columns for all nonzeros of A ($G(A)$),
- add fill edge $i \rightarrow j$ if there is a path from i to j in $G(A)$ through lower numbered vertices.

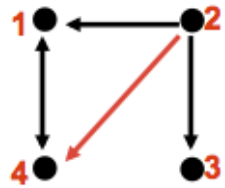
Fact: $G(L + U) = G(A)$, ignoring cancellations



A=LU



G(A)



$G^+(A)$

Sparse LU factorization with partial pivoting

Compute $P_r A P_c = LU$ where:

- A is large, sparse, nonsymmetric
- Columns reordered to reduce fill-in
- Rows reordered during the factorization by partial pivoting

Observations

– Symbolic and numeric factorizations are interleaved.

- A structure prediction step allows to compute upper bounds of the structure of L and U.
- These bounds are tight for strong Hall matrices (irreducible matrices which cannot be permuted to block upper triangular forms).

Structure prediction for sparse LU factorization

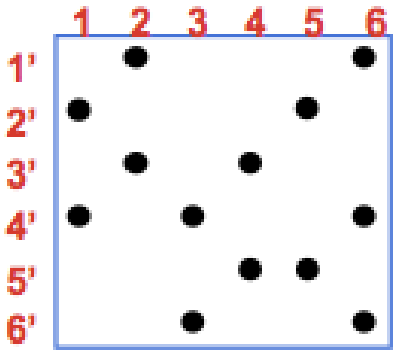
1. Compute an upper bound for the structure of L and U

Filled column intersection graph $G_\cap^+(A)$: Cholesky factor of $A^T A$

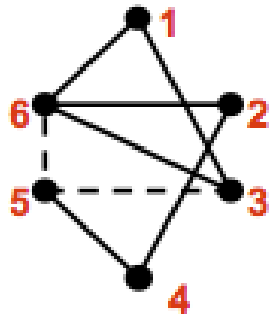
- $G(U) \subseteq G_\cap^+(A)$ and $G(L) \subseteq G_\cap^+(A)$

2. Predict potential dependencies between column computations

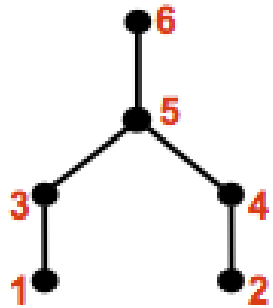
Column elimination tree $T_\cap(A)$: spanning tree of $G_\cap^+(A)$.



A



$G_\cap^+(A)$



$T_\cap(A)$

Different pivoting strategies

- Challenging to obtain good performance for sparse LU with partial pivoting on distributed memory computers
  - dynamic data structures
  - dependencies over-estimated
- Many problems can be solved with restricted/no pivoting and a few steps of iterative refinement.

$\Rightarrow$ motivation for SuperLU_DIST which implements LU with static pivoting

SuperLU_DIST [Li and Demmel, 2003]

1. Scale and permute A to maximize diagonal

$$A_1 = P_r D_r A D_c$$

2. Order equations and variables to reduce fill-in

$$A_2 = P_2 A_1 P_2^T$$

3. Symbolic factorization.
   - Identify supernodes, set up data structures and allocate memory for L, U.
4. Numerical factorization - LU with static pivoting
   - During the factorization $A_2 = LU$, replace tiny pivots by $\sqrt{\epsilon}||A||$
5. Triangular solutions - usually less than 5% total time.
6. If needed, use a few steps of iterative refinement

Symbolic factorization

- Complexity of symbolic factorization:
  - Greater than $nnz(L+U)$, but much smaller than $flops(LU)$.
  - No nice theory as in the case of symmetric matrices/chordal graphs.
  - Any algorithm which computes the transitive closure of a graph can be used.
- Why it is difficult to parallelize?
  - Algorithm is inherently sequential.
  - Small computation to communication ratio.
- Why do we need to parallelize ?
  - Memory needs = matrix A plus the factors L and U

$\Rightarrow$ Memory bottleneck for very large matrices

Parallel symbolic factorization [Grigori et al., 2007]

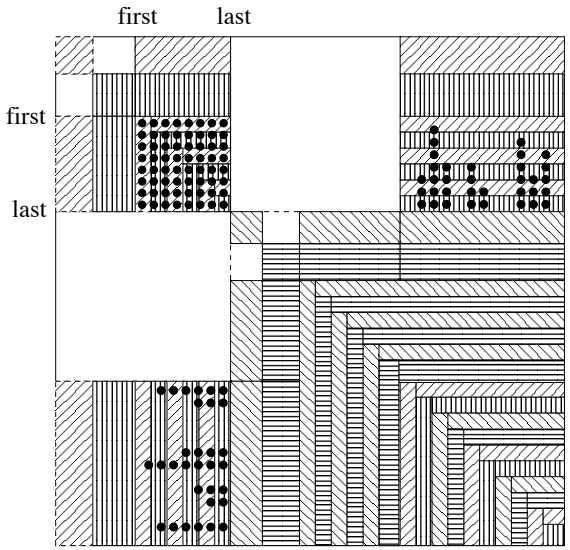
Goals

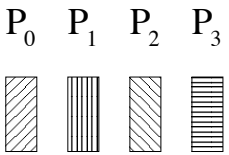
- Decrease the memory needs.
- Prevent this step from being a computational bottleneck of the factorization.

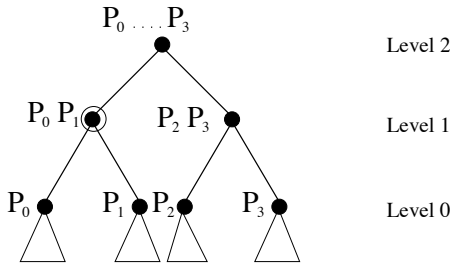
Approach

- Use a graph partitioning approach to partition the matrix.
- Exploit parallelism given by this partition and by a block cyclic distribution of the data.
- Identify dense separators, dense columns of L and rows of U to decrease computation.

Matrix partition and distribution







Separator tree - Exhibits computational dependencies

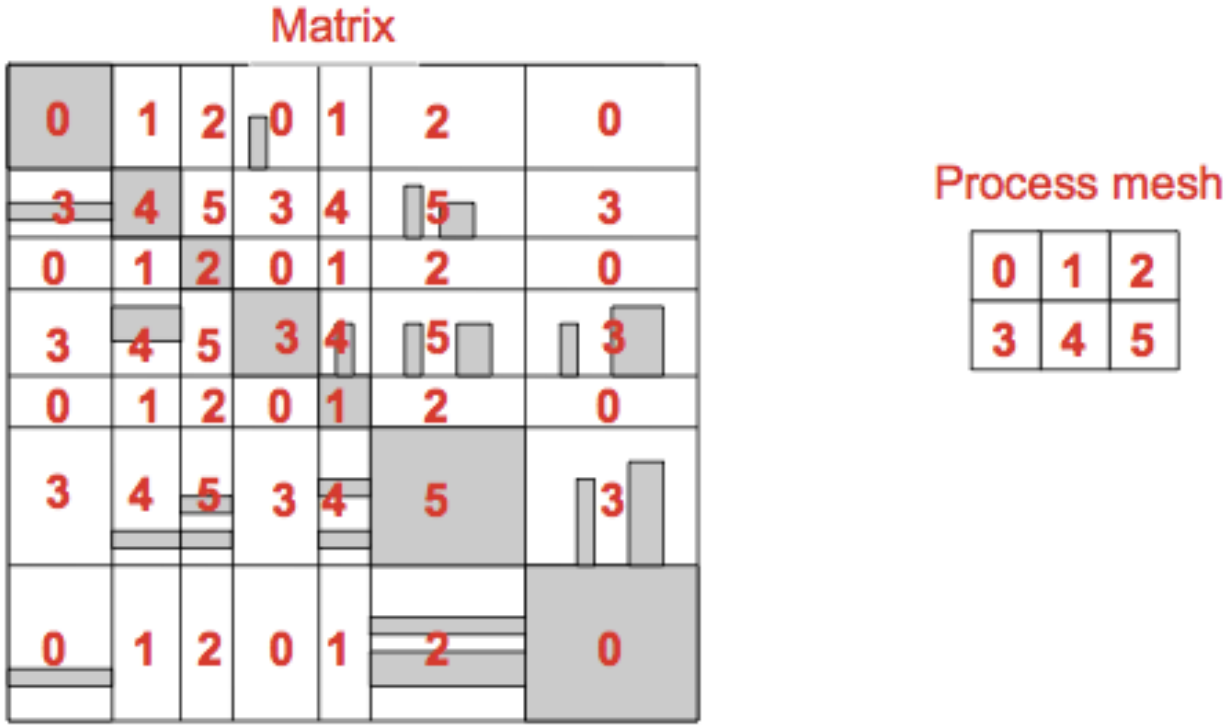
If node j updates node k, then j belongs to subtree rooted at k.

Algorithm

1. Assign all the processors to the root.
2. Distribute the root (1D block cyclic along the diagonal) to processors in the set.
3. Assign to each subtree half of the processors.
4. Go to Step 1 for each subtree which is assigned more than one processor.

Numeric factorization

- Supernodes (dense submatrices in L and U).
- Static pivoting (GESP) + iterative refinement.
- Parallelism from 2D block cyclic data distribution, pipelined right looking factorization.



Process mesh

0	1	2
3	4	5

SuperLU_DIST 2.5 and 3.0 on Cray XE6

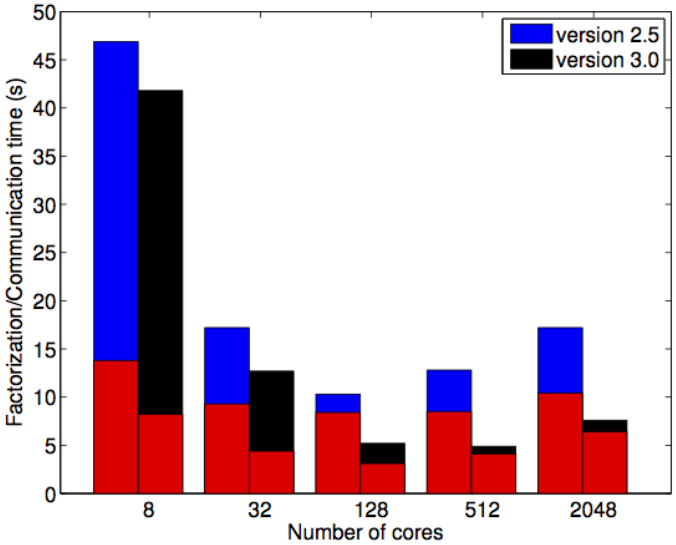


Figure : Accelerator, n=2.7M, fill=12x

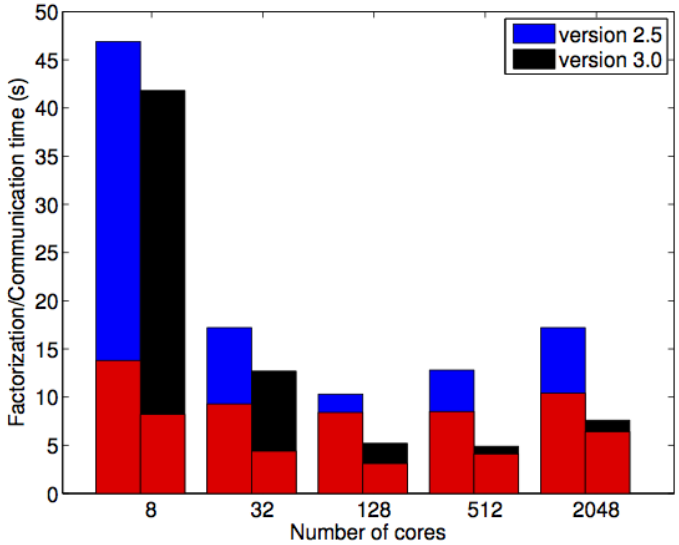


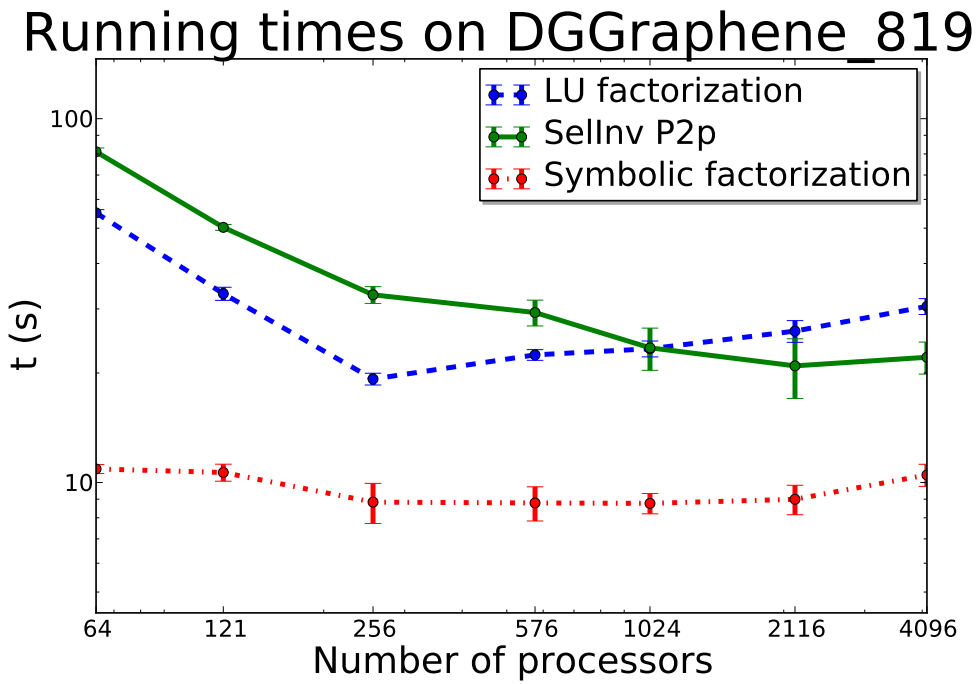
Figure : DNA, n = 445K, fill= 609x

- Version 2.5 - described in this lecture
- Version 3.0 - uses scheduling during numeric factorization
- Red part - computation time
- Blue/black part - communication/idle time
- Scheduling leads to up to 2.6 speedup

Courtesy of X. S. Li, LBNL

Experimental results (contd)

- Computing selected elements of A^{-1} by using LU factorization
- Electronic structure theory, disordered graphene system with 8192 atoms
- $n = 331K, nnz(A) = 349M, nnz(L) = 2973M$
- Edison: Cray XC30, 24 cores - two Intel Ivy Bridge procs per node, 1 MPI process per core, square grid of procs.



Courtesy of M. Jacquelin, `arXiv:1404.0447`

References (1)

Ballard, G., Buluc, A., Demmel, J., Grigori, L., Schwartz, O., and Toledo, S. (2013).

Communication optimal parallel multiplication of sparse random matrices.
In *In Proceedings of ACM SPAA, Symposium on Parallelism in Algorithms and Architectures*.

Duff, I. S. (1982).

Full matrix techniques in sparse gaussian elimination.
In Springer-Verlag, editor, *Lecture Notes in Mathematics (912)*, pages 71–84.

George, A. (1973).

Nested dissection of a regular finite element mesh.
SIAM Journal on Numerical Analysis, 10:345–363.

George, A., Liu, J. W.-H., and Ng, E. G. (1989).

Communication results for parallel sparse Cholesky factorization on a hypercube.
Parallel Computing, 10(3):287–298.

Gilbert, J. R. and Peierls, T. (1988).

Sparse partial pivoting in time proportional to arithmetic operations.
SIAM J. Sci. and Stat. Comput., 9(5):862–874.

Golub, G. H. and Van Loan, C. F. (2012).

Matrix Computations.
Johns Hopkins University Press, 4th edition.

Grigori, L., David, P.-Y., Demmel, J., and Peyronnet, S. (2010).

Brief announcement: Lower bounds on communication for direct methods in sparse linear algebra.
Proceedings of ACM SPAA.

References (2)

Grigori, L., Demmel, J., and Li, X. S. (2007).

Parallel symbolic factorization for sparse LU factorization with static pivoting.
SIAM Journal on Scientific Computing, 29(3):1289–1314.

Gupta, A., Karypis, G., and Kumar, V. (1995).

Highly scalable parallel algorithms for sparse matrix factorization.
IEEE Transactions on Parallel and Distributed Systems, 8(5).

Li, X. S. and Demmel, J. W. (2003).

SuperLU_DIST: A Scalable Distributed-memory Sparse Direct Solver for Unsymmetric linear systems.
ACM Trans. Math. Software, 29(2).

Liu, J. W. H. (1990).

The role of elimination trees in sparse factorization.
SIAM. J. Matrix Anal. & Appl., 11(1):134 – 172.

N.J.Higham (2002).

Accuracy and Stability of Numerical Algorithms.
SIAM, second edition.

Parter, S. (1961).

The use of linear graphs in gaussian elimination.
SIAM Review, pages 364–369.

Rose, D. J. (1970).

Triangulated graphs and the elimination process.
Journal of Mathematical Analysis and Applications, pages 597–609.

References (3)

Rose, D. J. and Tarjan, R. E. (1978).

Algorithmic aspects of vertex elimination on directed graphs.
SIAM J. Appl. Math., 34(1):176–197.

Schreiber, R. (1982).

A new implementation of sparse gaussian elimination.
ACM Trans. Math. Software, 8:256–276.