Factorization-based Sparse Solvers and Preconditioners

X. Sherry Li
xsli@lbl.gov
Lawrence Berkeley National Laboratory

SIAM Annual Meeting, July 12-16, 2010, Pittsburgh

Acknowledgements

- **Collaborators**
  - Ming Gu, University of California, Berkeley
  - Esmond Ng, Lawrence Berkeley National Lab
  - Meiyue Shao, Umeå University, Sweden
  - Panayot Vassilevski, Lawrence Livermore National Lab
  - Jianlin Xia, Purdue University
  - Ichitaro Yamazaki, Lawrence Berkeley National Lab

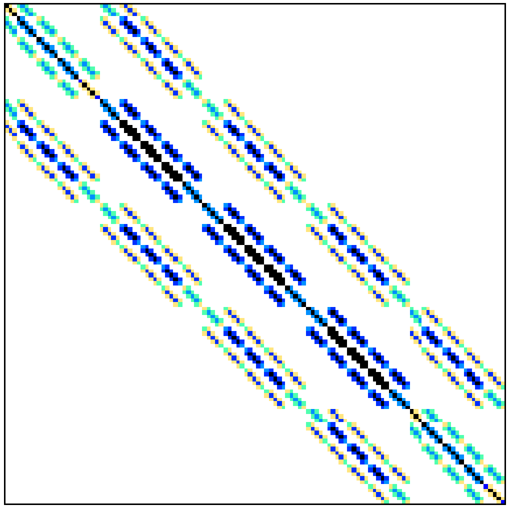
- **Funded through DOE SciDAC projects**
  - TOPS (Towards Optimal Petascale Simulations)
  - CEMM (Center for Extended MHD Modeling)
  - ComPASS (Community Petascale Project for Accelerator Science and Simulation)

The Problem

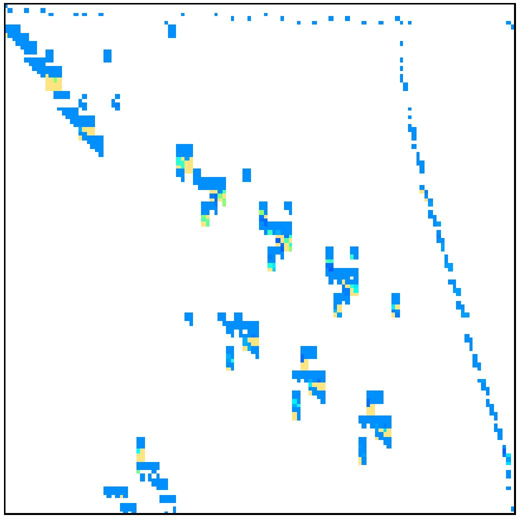
Solve Ax = b, A is sparse, b is dense or sparse

- Example: A of dimension 10^6,   10~100 nonzeros per row
- fluid dynamics, structural mechanics, chemical process simulation, circuit simulation, electromagnetic fields, magneto-hydrodynamics, seismic-imaging, economic modeling, optimization, data analysis, statistics, . . .

Boeing/msc00726



Mallya/lhr01



The algorithm . . . factorization

- **Gaussian elimination: A = LU**
  - A is modified . . . numerically as well as pattern-wise

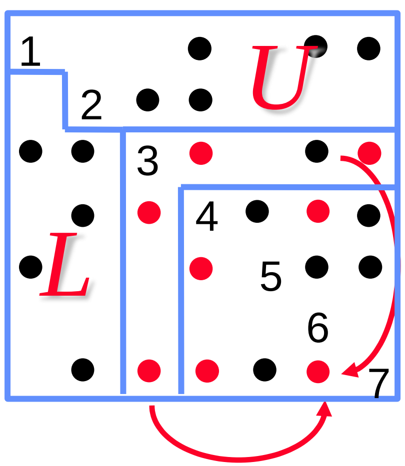
- **Deliver reliable solution, error bounds, condition estimation, multiple RHS, . . .**

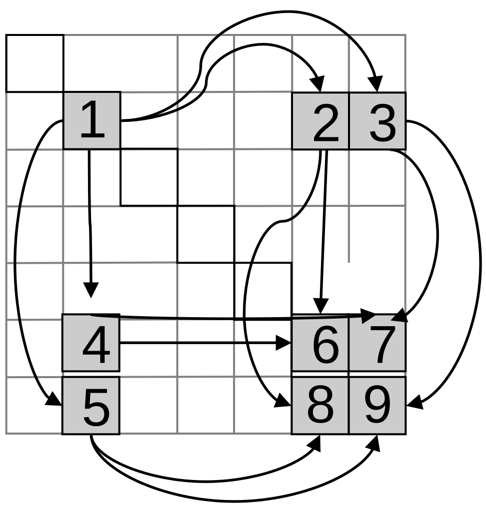
- **Complexity wall**

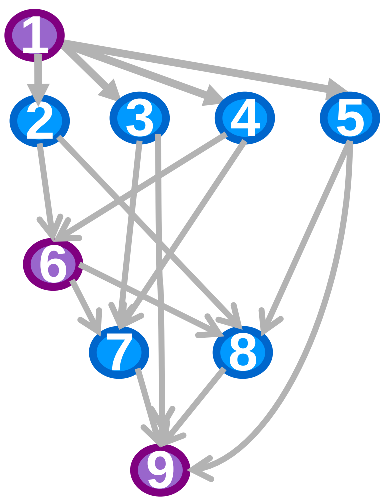
  <u>Theorem:</u> for model problems, Nested Dissection ordering gives optimal complexity in exact arithmetic [George '73, Hoffman/Martin/Rose, Eisenstat, Schultz and Sherman]
  - 2D (kxk = N grids): O(N logN) memory, $O(N^{3/2})$ operations
  - 3D (kxkxk = N grids): $O(N^{4/3})$ memory, $O(N^2)$ operations

Sparse factorization

- **Store A explicitly … many sparse compressed formats**
- **"Fill-in" . . . new nonzeros in L & U**
- **Graph algorithms: directed/undirected graphs, bipartite graphs, paths, elimination trees, depth-first search, heuristics for NP-hard problems, cliques, graph partitioning, . . .**
- **Unfriendly to high performance, parallel computing**
  - Irregular memory access, indirect addressing, strong task/data dependency







Available direct solvers

- **Survey of different types of factorization codes**

http://crd.lbl.gov/~xiaoye/SuperLU/SparseDirectSurvey.pdf

  - LL^T (s.p.d.)
  - LDL^T (symmetric indefinite)
  - LU (nonsymmetric)
  - QR (least squares)
  - Sequential, shared-memory (multicore), distributed-memory, out-of-core

- **Our work focuses on unsymmetric LU**
  - Sequential SuperLU [Demmel/Eisenstat/Gilbert/Liu/L. '99]
  - SuperLU_MT [L./Demmel/Gilbert '99] : Pthreads, OpenMP
  - SuperLU_DIST [L./Demmel/Grigori '00] : MPI

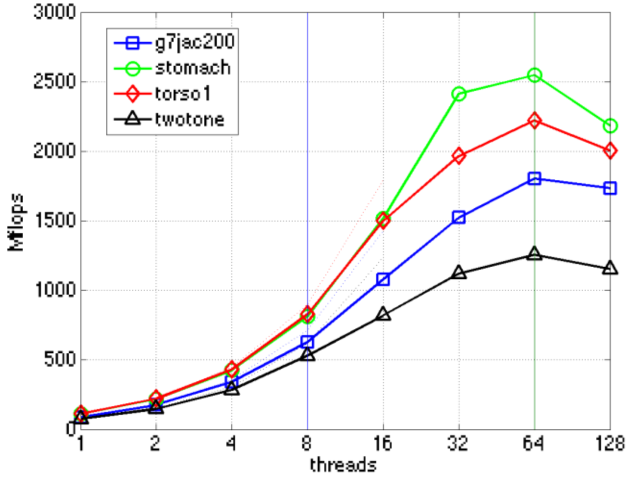
How useful?

- **Download counts**

	FY 2006	FY 2009
Total	6176	9983
SuperLU	4361	5719
SuperLU_MT	690	1779
SuperLU_DIST	1125	2485

- **Sun VictoriaFalls: MC+MT**
  - 1.4 GHz UltraSparc T2
    1.4 Gflops/core
  - 2 sockets
    8 cores/socket
    8 hardware threads/core
  - ➢ Maximum speedup 20
    effective use of 64 threads

SuperLU_MT



Beyond direct solver

- **Factorization variants very useful for constructing preconditioners for an iterative solver**
  - Approximate factorization: Incomplete LU (ILU), approximate inverse, …
  - Factorization of subproblems: Schur complement method …

Rest of the talk . . .

- **Supernodal ILU**
  - Available in SuperLU 4.0
- **Hybrid solver based on Schur complement method**
- **Rank structured sparse factorization**

ILU preconditioner

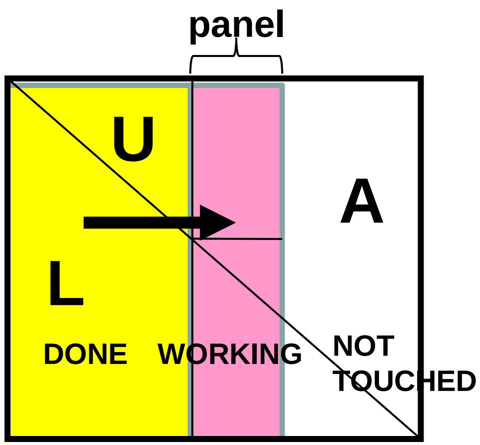
- **Structure-based dropping: level-of-fill**
  - ILU(0), ILU(1), …
  - Rationale: the higher the level, the smaller the entries
  - Separate symbolic factorization to determine fill-in pattern
- **Value-based dropping: drop truly small entries**
  - Fill-in pattern determined on-the-fly

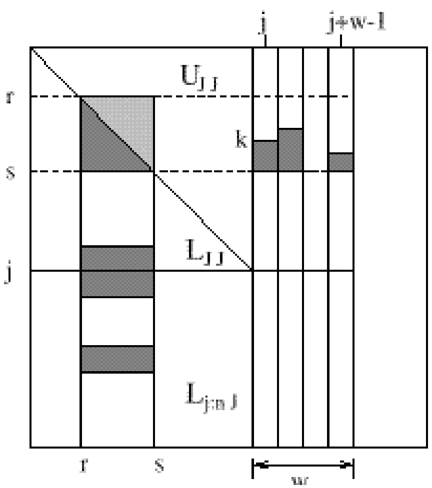
- **ILUTP** [Saad]**: among the most sophisticated, and (arguably) robust; implementation similar to direct solver**
  - "T" = threshold, "P" = pivoting
  - Dual dropping: ILUTP(p, $\mathcal{T}$)
    - **Remove elements smaller than $\mathcal{T}$**
    - **At most p largest kept in each row or column**

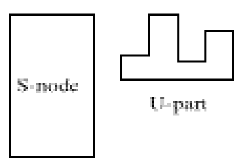
SuperLU [Demmel/Eisenstat/Gilbert/Liu/L. '99]

http://crd.lbl.gov/~xiaoye/SuperLU

- **Left-looking, supernode**







1. Sparsity ordering of columns
   use graph of A'*A
2. Factorization
   For each panel …
   - Partial pivoting
   - Symbolic fact.
   - Num. fact. (BLAS 2.5)
3. Triangular solve

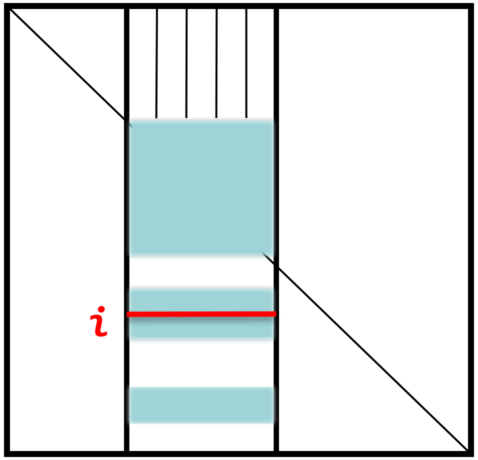
Primary dropping rule: S-ILU(τ)

- **Similar to ILUTP, adapted to supernode**
  1. U-part:

     $$\text{If } |u_{ij}| < \tau \cdot \|A(:,j)\|_\infty, \text{ then set } u_{ij} = 0$$

  2. L-part: retain supernode

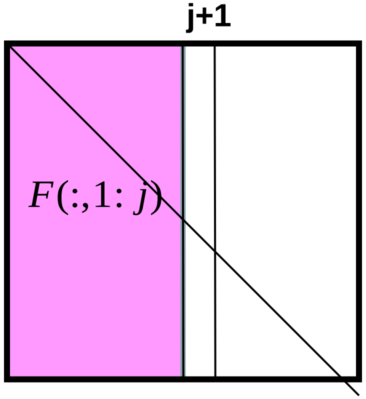
     $$\text{Supernode } L(:,s:t), \text{if } RowSize(i, s:t) < \tau, \text{then set the entire } i\text{-th row to zero}$$



- **Remarks**
  1) Delayed dropping
  2) Entries computed first, then dropped.
     May not save many flops compared to LU
  3) Choices for RowSize() metric
     e.g., $RowSize(x) = \| x \|_\infty$

Secondary dropping rule: S-ILU(p, $\mathcal{T}$)

- **Control fill ratio with a user-desired upper bound γ**
- **Earlier work, column-based**
  - [Saad]: ILU(p, $\mathcal{T}$), at most p largest nonzeros allowed in each row
  - [Gupta/George]: p adaptive for each column $p(j) = \gamma \cdot nnz(A(:,j))$



- **Our new scheme is area-based**
  - Look at fill ratio from column 1 up to j:

    $$fr(j) = nnz(F(:,1:j)) / nnz(A(:,1:j))$$

  - Define adaptive upper bound function $f(j) \in [1, \gamma]$

    If $fr(j)$ exceeds $f(j)$, retain only p largest, such that $fr(j) \le f(j)$

➢ More flexible, allow some columns to fill more, but limit overall

Experiments: GMRES + ILU

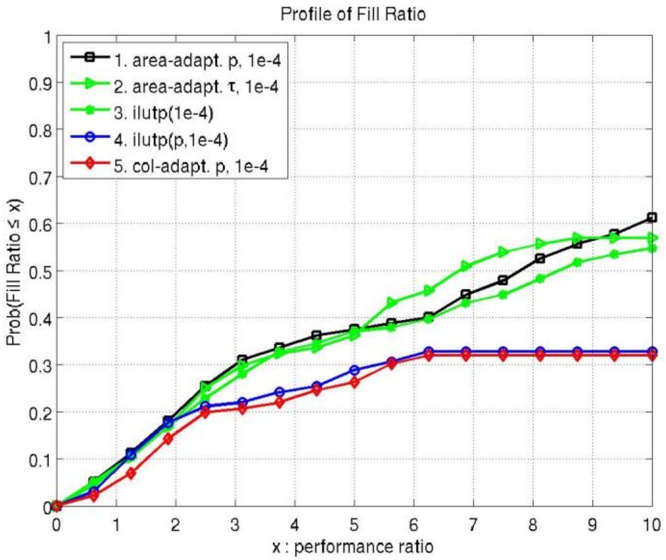
- **232 unsymmetric test matrices**
  RHS is generated so the true solution is 1-vector
  - 227 from Univ. of Florida Sparse Matrix Collection, dimension 5K–1M, condition number below 10^{15}
  - 5 from MHD calculation in tokmak design in fusion plasma

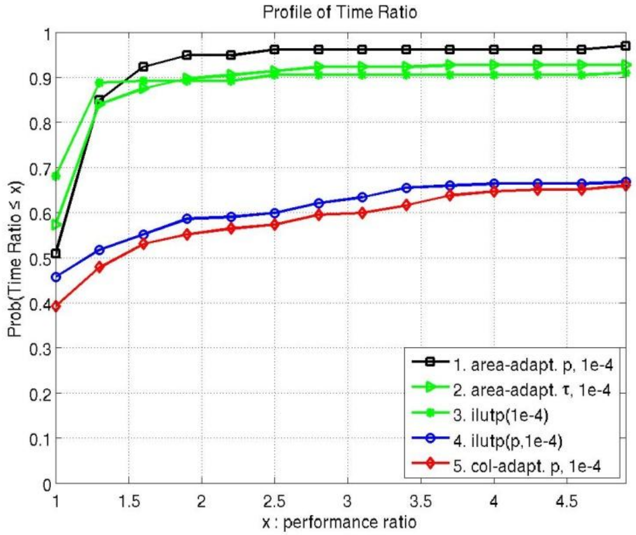
- **Use restarted GMRES with ILU as a right preconditioner**
  Solve $PA(\tilde{L}\tilde{U})^{-1}y = Pb$
  - Size of Krylov subspace set to 50
  - Initial guess is a 0-vector
  - Stopping criteria: $\|b\text{-}Ax_k\|_2 \le 10^{-8}\|b\|_2$ and ≤ 500 iterations

- **AMD Opteron 2.4 GHz quad-core (Cray XT5), 16 GBytes memory, PathScale pathcc and pathf90 compilers**

S-ILU comprehensive tests

- **Performance profile of fill ratio – fraction of the problems a solver could solve within a fill ratio of X**
- **Performance profile of runtime – fraction of the problems a solver could solve within a factor X of the best solution time**





- **Conclusion:**
  - New area-based heuristic is much more robust than column-based one
  - ILUTP($\mathcal{T}$) is reliable; but need secondary dropping to control memory

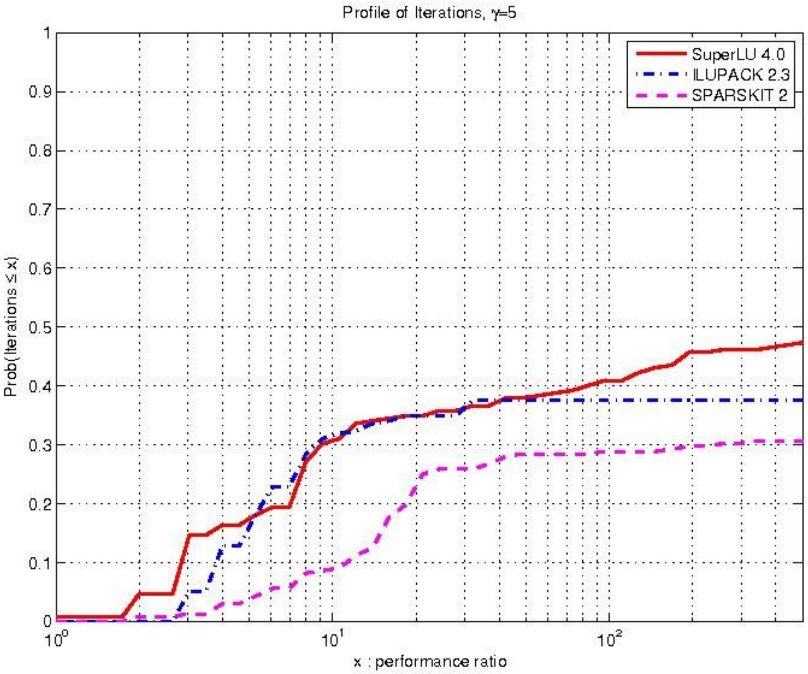
Compare with the other preconditioners

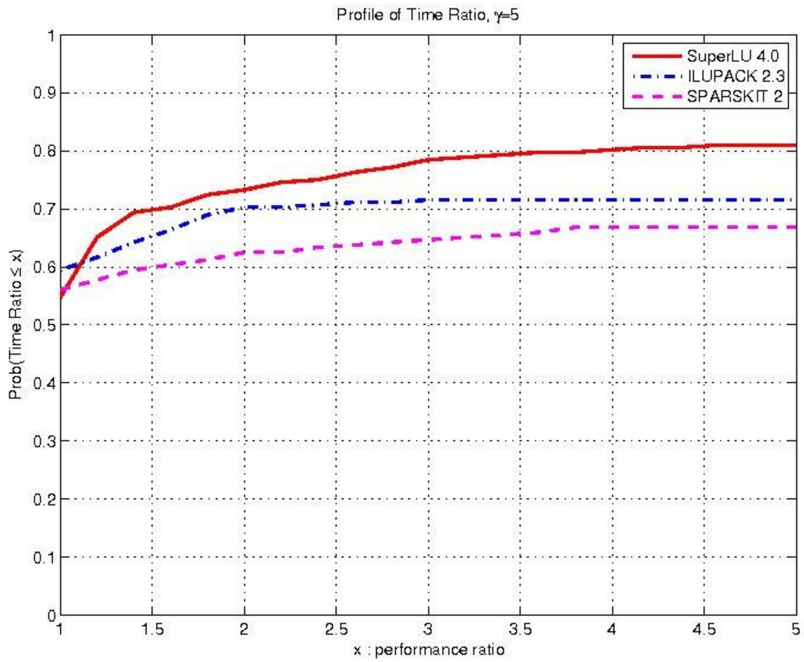
- **SPARSKIT** [saad] **: ILUTP, closest to ours**
  - Row-wise algorithm, no supernode
  - Secondary dropping uses a fixed p for each row

- **ILUPACK** [Bolhoefer et al.] : **very different**
  - Inverse-based approach: monitor the norm of the k-th row of L^{-1}, if too large, delay pivot to next level
  - Multilevel: restart the delayed pivots in a new level

Compare with SPARSKIT, ILUPACK

- **S-ILU:** $\tau = 10^{-4}, \gamma = 5,\ \text{diag\_thresh}\ \eta = 0.1$
- **ILUPACK :** $\tau = 10^{-4},\ \gamma = 5,\ \nu = 5$
- **SPARSKIT :** $\tau = 10^{-4},\ \gamma = 5,\ p = \gamma \cdot \dfrac{nnz}{n}$





Comparison (cont) … a closer look …

- **S-ILU and ILUPACK are comparable: S-ILU is slightly faster, ILUPACK has slightly lower fill**
- **No preconditioner works for all problems . . .**
- **They do not solve the same set of problems**
  - S-ILU succeeds with 142
  - ILUPACK succeeds with 130
  - Both succeed with 100 problems

➢ **Two methods complimentary to one another, both have their place in practice**

Schur complement method

- **a.k.a iterative substructuring method**
  or, non-overlapping domain decomposition

- **Divide-and-conquer paradigm . . .**
  - Divide entire problem (domain, graph) into subproblems (subdomains, subgraphs)
  - Solve the subproblems
  - Solve the interface problem (Schur complement)

- **Variety of ways to solve subdomain problems and the Schur complement … lead to a powerful polyalgorithm or hybrid solver framework**

Algebraic view

1. **Reorder into 2x2 block system, A_{11} is block diagonal**

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

2. **Schur complement**

$$S = A_{22} - A_{21} A_{11}^{-1} A_{12} = A_{22} - (U_{11}^{-T} A_{21}^{T})^{T} (L_{11}^{-1} A_{12}) = A_{22} - W \cdot G$$

$$\text{where} \quad A_{11} = L_{11} U_{11}$$

S corresponds to interface (separator) variables, no need to be formed explicitly

3. **Compute the solution**

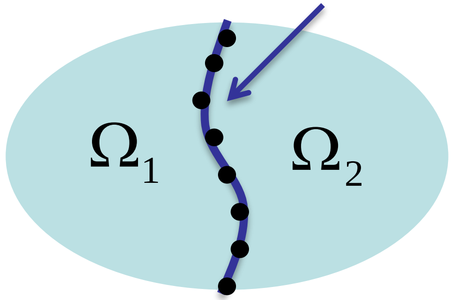
$$(1) \quad x_2 = S^{-1}(b_2 - A_{21} A_{11}^{-1} b_1) \quad \leftarrow \text{iterative solver}$$

$$(2) \quad x_1 = A_{11}^{-1}(b_1 - A_{12} x_2) \quad \leftarrow \text{direct solver}$$

Structural analysis view

- **Case of two subdomains**

Substructure contribution: $A^{(i)} = \begin{pmatrix} A_{i\,i}^{(i)} & A_{i\,I}^{(i)} \\ A_{I\,i}^{(i)} & A_{I\,I}^{(i)} \end{pmatrix}$ $\quad i = \text{"interior"}$, $\quad I = \text{"Interface"}$



1. Assembled block matrix $A = \begin{pmatrix} A_{i\,i}^{(1)} & & A_{i\,I}^{(1)} \\ & A_{i\,i}^{(2)} & A_{i\,I}^{(2)} \\ A_{I\,i}^{(1)} & A_{I\,i}^{(2)} & A_{I\,I}^{(1)} + A_{I\,I}^{(2)} \end{pmatrix}$

2. Perform direct elimination of $A^{(1)}$ and $A^{(2)}$ independently,

   Local Schur complements: $S^{(i)} = A_{I\,I}^{(i)} - A_{I\,i}^{(i)} A_{i\,i}^{(i)^{-1}} A_{i\,I}^{(i)}$

   Assembled Schur complement $S = S^{(1)} + S^{(2)}$

Solving the Schur complement system

- **Proposition** [Smith/Bjorstad/Gropp'96]

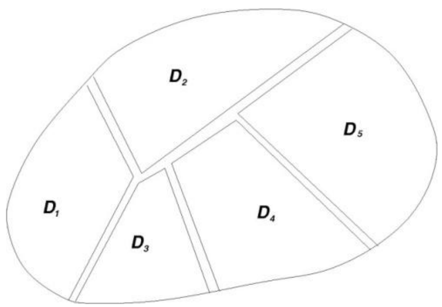
  For an SPD matrix, condition number of a Schur complement is no larger than that of the original matrix.
- **S is much reduced in size, better conditioned, but denser**
  - solvable with preconditioned iterative solver

- **Two approaches to preconditioning S**
  1. Explicit S (e.g., HIPS [Henon/Saad'08], and ours)
     - **can construct general algebraic preconditioner, e.g. ILU(S), must preserve sparsity of S**
  2. Implicit S (e.g. [Giraud/Haidary/Pralet'09])
     - **preconditioner construction is restricted; more parallel**
     - **E.g., additive Schwarz preconditioner** $S = S^{(1)} \oplus S^{(2)} \oplus S^{(3)} \ldots$

$$M = {S^{(1)}}^{-1} \oplus {S^{(2)}}^{-1} \oplus {S^{(3)}}^{-1} \ldots$$

Parallelism – extraction of multiple subdomains

- **Partition adjacency graph of |A|+|A^T|**
  Goals: reduce size of separator, balance subdomains sizes
  - nested dissection (e.g., PT-Scotch, ParMetis)
  - k-way partition (preferred)

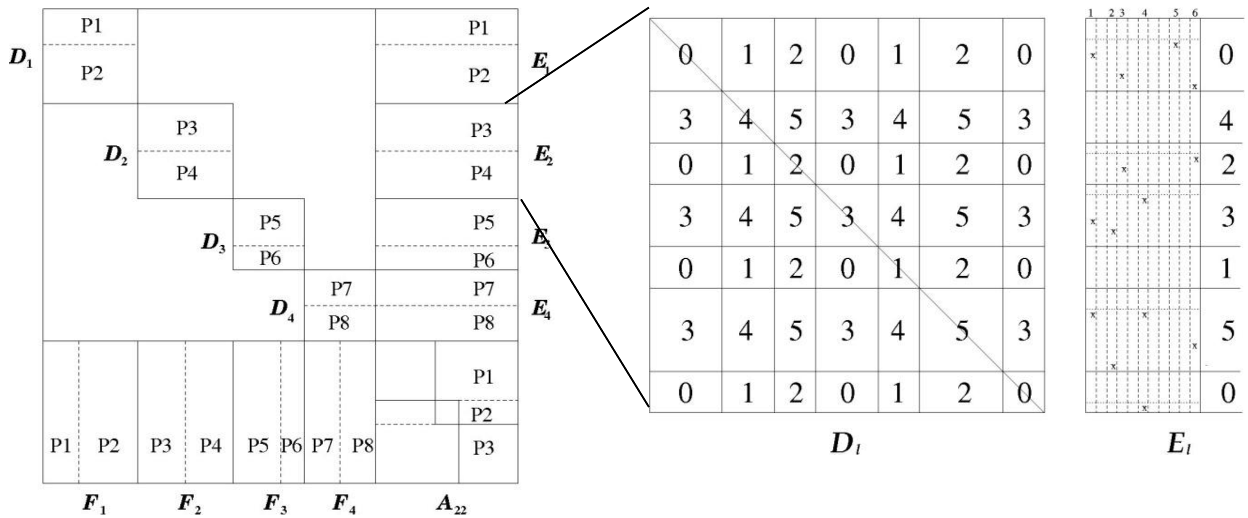


$$\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right) = \left(\begin{array}{cccc|c} D_1 & & & & E_1 \\ & D_2 & & & E_2 \\ & & \ddots & & \vdots \\ & & & D_k & E_k \\ \hline F_1 & F_2 & \dots & F_k & A_{22} \end{array}\right)$$

- **Memory requirement: fill is restricted within**
  - "small" diagonal blocks of A_{11}, and
  - ILU(S), sparsity can be enforced

Hierarchical parallelism

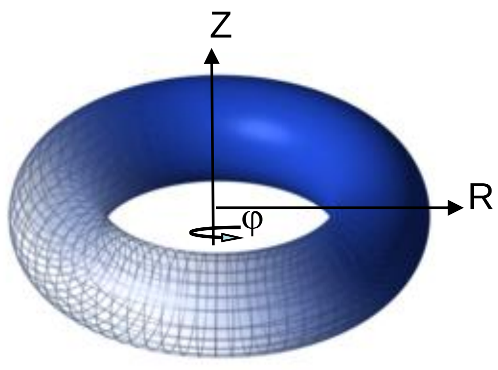
- **Multiple procs per subdomain**
  - one subdomain with 2x3 procs (e.g. SuperLU_DIST, MUMPS)



- **Advantages:**
  - Only need modest level of parallelism from direct solver.
  - Can keep fixed and modest number of subdomains when increasing processor count. The size of the Schur complement system is constant, and convergence rate is constant, regardless of processor count.

Application 1: Burning plasma for fusion energy

- DOE SciDAC project: Center for Extended Magnetohydrodynamic Modeling (CEMM), PI: S. Jardin, PPPL
- **Develop simulation codes for studying the nonlinear macroscopic dynamics of MHD-like phenomena in magnetized fusion plasmas in a tokamak, address critical issues facing burning plasma experiments such as ITER**
- **Simulation code suite includes M3D-C^1, NIMROD**



[S. Jardin]

- At each φ = constant plane, scalar 2D data is represented using 18 degree of freedom quintic triangular finite elements Q_{18}
- Coupling along toroidal direction

S-ILU for extended MHD (fusion)

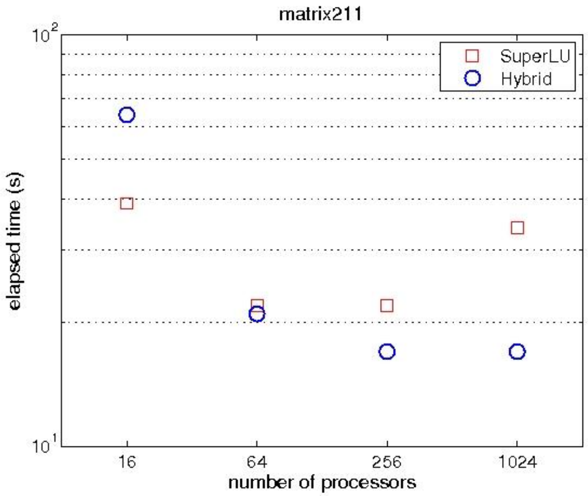
- **ILU parameters:** $\tau = 10^{-4}, \gamma = 10$
- **Matrices from M3D-C1 simulation code**

Problems	order	Nonzeros (millions)	SuperLU Time	fill-ratio	ILU time	fill-ratio	GMRES Time	Iters
matrix31	17,298	2.7 m	33.3	13.1	8.2	2.7	0.6	9
matrix41	30,258	4.7 m	111.1	17.5	18.6	2.9	1.4	11
matrix61	66,978	10.6 m	612.5	26.3	54.3	3.0	7.3	20
matrix121	263,538	42.5 m	x	x	145.2	1.7	47.8	45
matrix181	589,698	95.2 m	x	x	415.0	1.7	716.0	289

- **Up to 9x smaller fill ratio, and 10x faster**

Hybrid solver for extended MHD (fusion)

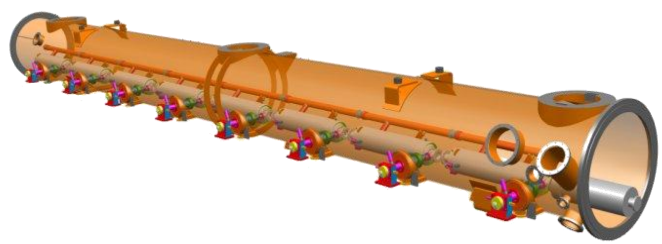
- **Cray XT4 at NERSC**
- **Matrix211 – dimension = 801K, nonzeros = 129M, real, unsymmetric, indefinite**
  - PT-Scotch extracts 8 subdomains of size ≈ 99K, S of size ≈ 13K
  - SuperLU_DIST to factorize each subdomain, and compute preconditioner LU($\tilde{S}$)
  - BiCGStab of PETSc to solve Schur system on 64 processors with residual < 10^{-12} , converged in 10 iterations
- **Needs only 1/3 memory of direct solver**



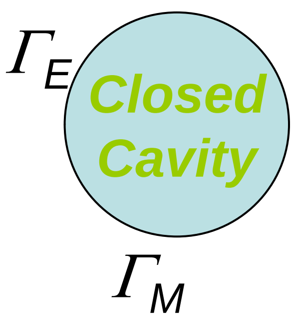
Application 2: Accelerator cavity design

- DOE SciDAC: Community Petascale Project for Accelerator Science and Simulation (ComPASS), PI: P. Spentzouris, Fermilab
- **Development of a comprehensive computational infrastructure for accelerator modeling and optimization**
- **RF cavity: Maxwell equations in electromagnetic field**
- **FEM in frequency domain leads to large sparse eigenvalue problem; needs to solve shifted linear systems**

[L.-Q. Lee]

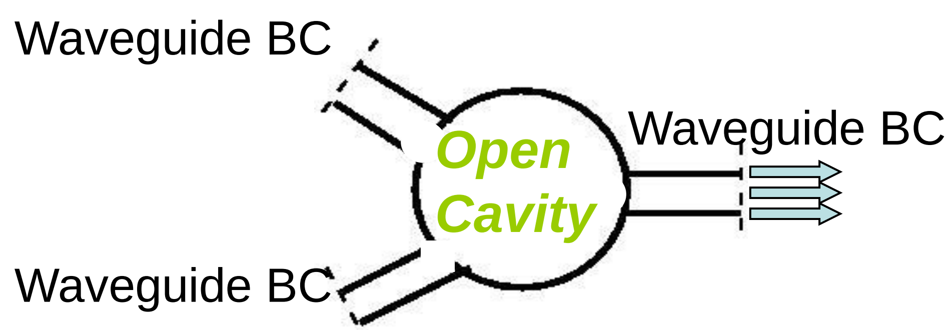


RF unit in ILC



linear eigenvalue problem

$$(K_0 - \sigma^2 M_0)\, x = M_0\, b$$

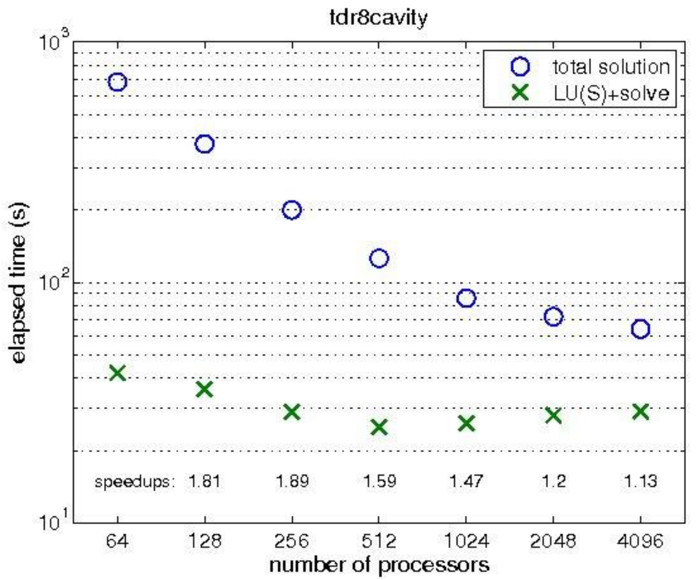


nonlinear complex eigenvalue problem

$$(K_0 + i\,\sigma W - \sigma^2 M_0)\, x = b$$

Hybrid solver for RF cavity design

- **Cray XT4 at NERSC**
- **Tdr8cavity – design for International Linear Collider**
  - dimension = 17.8M, nonzeros = 727M
  - PT-Scotch extracts 64 subdomains of size ≈ 277K, S of size ≈ 57K
  - BiCGStab of PETSc to solve Schur system on 64 processors with residual < 10^{-12}, converged in 9 – 10 iterations
- **Direct solver failed !**



Computing approximate Schur as preconditioner

Combinatorial problems . . .

- **Sparse triangular solution with many sparse RHSs**

$$S = A_{22} - \sum_l (U_l^{-T} F_l^T)^T (L_l^{-1} E_l), \text{ where } D_l = L_l U_l$$

- **Sparse matrix–sparse matrix multiplication**

$$\tilde{G} \leftarrow \text{sparsify}(G, \sigma_1); \quad \tilde{W} \leftarrow \text{sparsify}(W, \sigma_1)$$

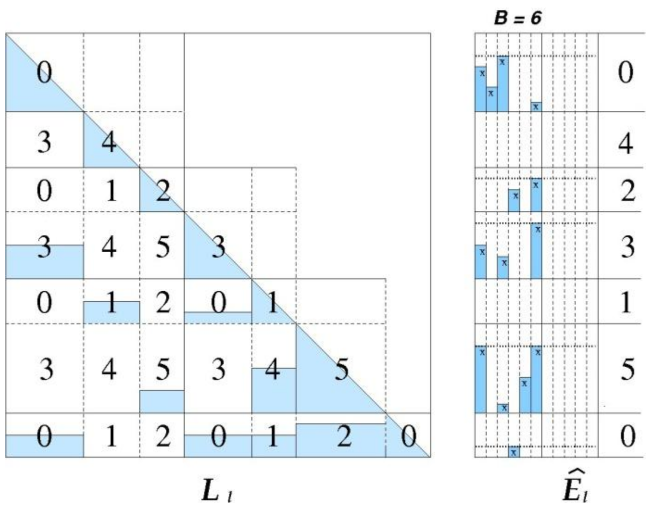
$$T^{(p)} \leftarrow \tilde{W}^{(p)} \times \tilde{G}^{(p)}$$

$$\hat{S}^{(p)} \leftarrow A_{22}^{(p)} - \sum_q T^{(q)}(p); \quad \tilde{S} \leftarrow \text{sparsify}(\hat{S}, \sigma_2)$$

- **K-way graph partitioning with multiple constraints**
  - Small separator
  - Similar subdomains
  - Similar connectivity

Sparse triangular solution with sparse RHSs

- **RHS vectors E_ℓ and F_ℓ are sparse (e.g., about 20 nnz per column); There are many RHS vectors (e.g., $O(10^4)$ columns)**

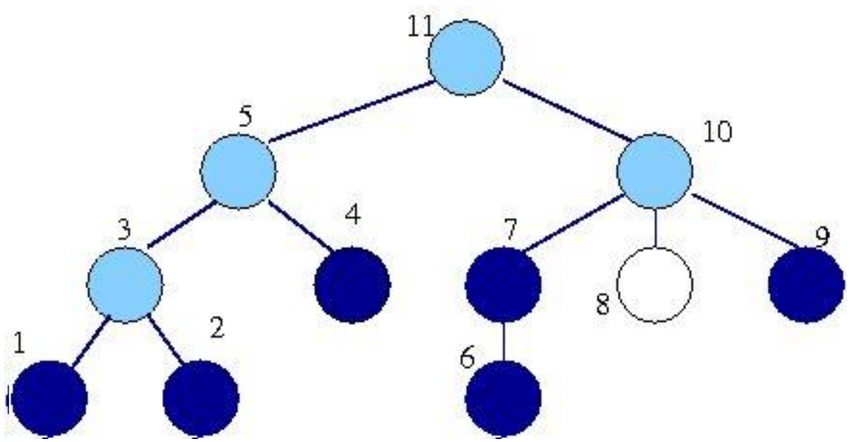


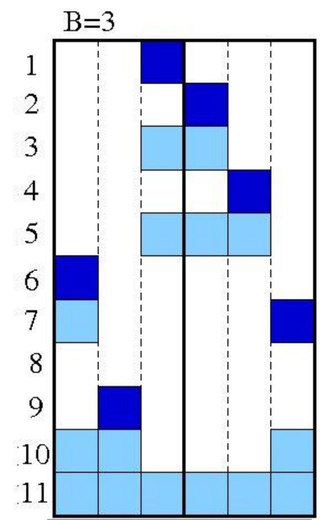
- **Blocking the RHS vectors**
  - Reduce number of calls to the symbolic routine and number of messages, and improve read reuse of the LU factors
  - ➢ Achieved over 5x speedup
  - ✖ zeros must be padded to fill the block

Sparse triangular solution with sparse RHSs

- **Combinatorial question: Reorder columns of E_ℓ to maximize structural similarity among the adjacent columns.**
- **Where are the fill-ins?**

  <u>Path Theorem</u> [Gilbert'94] **Given the elimination tree of D_l, fill will be generated in G_l at the positions associated with the nodes on the path from nodes of the nonzeros in E_l to the root**

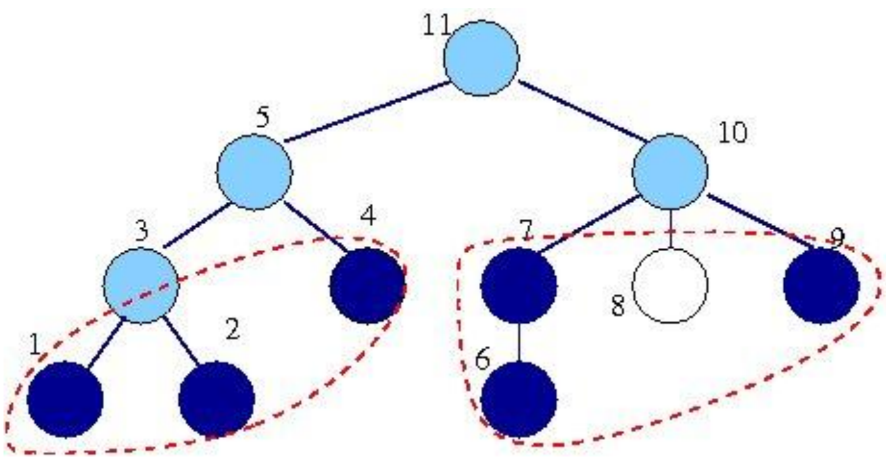


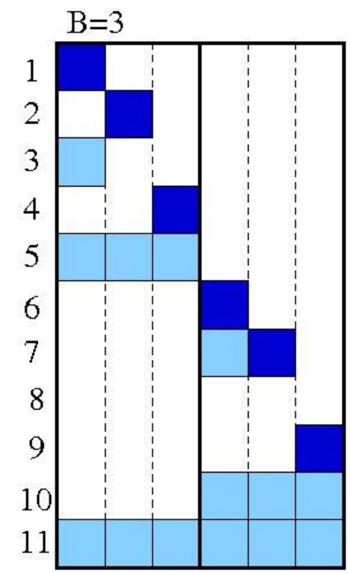


24 padded zeros

Sparse triangular solution … postordering

- **Postorder-conforming ordering of the RHS vectors**
  - Postorder the elimination tree
  - Permute the columns of E_I such that the row indices of the first nonzeros are in ascending order
- **Increased overlap of the paths to the root, fewer padded zeros**
- **30-60% speedup**





13 padded zeros

Sparse triangular solution … further optimization

- **A reordering based on a hyper-graph partitioning model which minimizes certain cost function that measures the dissimilarity of the sparsity pattern within a partition. This led to additional 10% speedup.**

Hybrid solver summary

- **Multiple levels of parallelism is essential for difficult problems and large core count.**

- **Tuning parameter:**

  Number of subdomains represents important trade-off between direct solver scalability and convergence rate of the iterative solver of the Schur system.

Forward looking . . .

- **Can we break the complexity wall of factorization?**
  - 2D (kxk = N grids): O(N logN) memory, $O(N^{3/2})$ operations
  - 3D (kxkxk = N grids): $O(N^{4/3})$ memory, $O(N^2)$ operations

- **. . . Combine rank structured factorization with sparsity structure → sparse structured factorization**

Rank structured matrices

- **Fast multipole method**
  - Greengard, Roklin, Starr, et al.
- **Hierarchical matrices: $\mathcal{H}$-matrix, $\mathcal{H}^2$-matrix**
  - Bebendorf, Börm, Grasedyck, Hackbusch, Le Borne, Martinsson, Tygert, et al.
- **Quasi-separable matrices**
  - Bini, Eidelman, Gemignani, Gohberg, Olshevsky, Van Barel, et al.
- **Semi-separable matrices**
  - Chandrasekaran, Dewilde, Gohberg, Gu, Kailath, Van Barel, van der Veen, Vandebril, White, et al.
- **Others . . .**

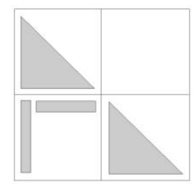
Rank structured dense Cholesky

- **One step of factorization**

$$F = \begin{pmatrix} L_{11} & \\ L_{21} & I \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ & S \end{pmatrix}$$

- **Data compression of off-diagonal block**
  - rank revealing QR or τ - accurate SVD

$$L_{21} = (U \quad U^T) \begin{pmatrix} \Sigma & \\ & \hat{\Sigma} \end{pmatrix} \begin{pmatrix} V^T \\ \hat{V}^T \end{pmatrix} = U\Sigma V^T + \hat{U}\hat{\Sigma}\hat{V}^T, \ \Sigma \text{ is of size } r, \ \| \hat{U}\hat{\Sigma}\hat{V}^T \|_2 = O(\tau)$$



- **Approximate factor**

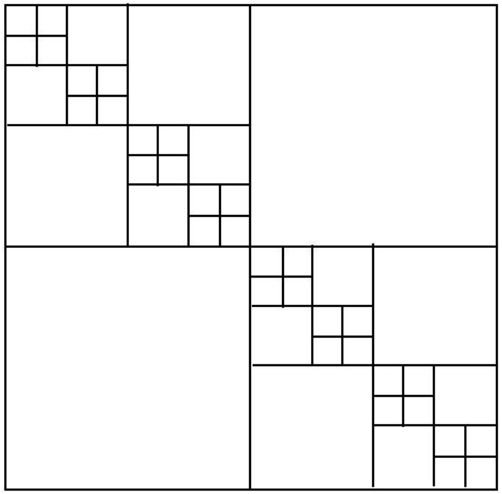
$$\text{approximate Schur} : \tilde{S} = F_{22} - U\Sigma^2 U^T = S + \hat{U}\hat{\Sigma}^2\hat{U}^T = S + O(\tau^2)$$

$$\tilde{F} = \tilde{L}\tilde{L}^T = F + \begin{pmatrix} 0 & 0 \\ 0 & \hat{U}\hat{\Sigma}^2\hat{U}^T \end{pmatrix} = F + \begin{pmatrix} 0 & 0 \\ 0 & O(\tau^2) \end{pmatrix}, \ \text{guaranteed SPD}$$

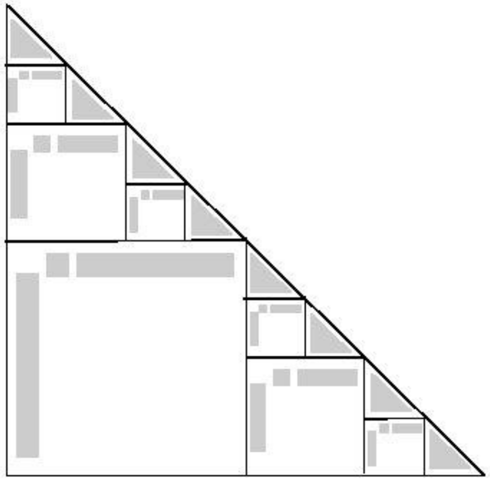
Multiple blocks

- **Hierarchical factorization**

Recursive partitioning



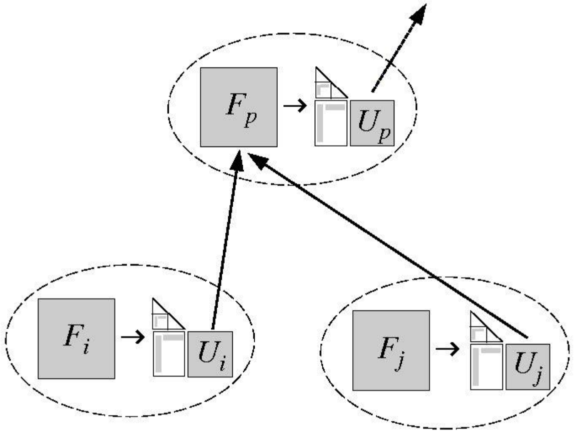
Structured factor



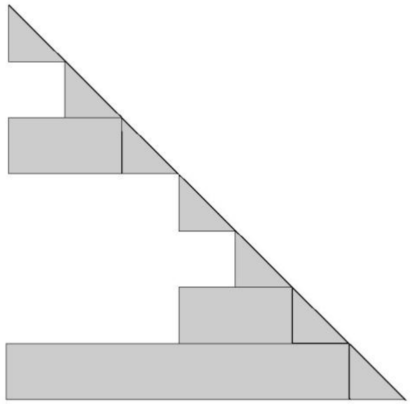
- **Complexity . . . almost linear !**
  - Factorization: $O(r\ N^2)$
  - Solution: $O(r\ N)$
  - Storage: $O(r\ N)$

Sparse structured factorization

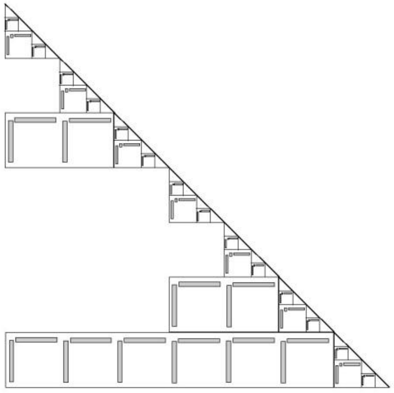
- **Low-rank property of the intermediate dense matrices**
  - Discretized PDEs: dense fill-in, Schur complements
- **Multifrontal factorization kernels**
  - Frontal matrices: F_i
  - Update matrices: U_i
  - ➢ Numerical ranks: 10 - 20



- **Nested dissection ordering**



Classical factor



Structured factor

Results of sparse structured MF factorization

- **Complexity**
  - Lower levels: standard factorization, upper levels: structured factorization
  - Classical factorization: $O(N^{3/2})$
  - Structured factorization: $O(r^2 N)$
- **Performance**
  - For 2D Model problem of mesh size 4096^2, as a direct solver, 10x faster than classical MF
  - For linear elasticity problems, as a preconditioner (with larger τ), the condition numbers of the preconditioned systems are small and essentially constant, independent of the λ/μ ratio.

$$-(\mu\Delta u + (\lambda+\mu)\nabla\nabla \bullet u) = f \text{ in } \Omega = (0,1)\times(0,1)$$

$$u = 0 \quad \text{on } \partial\Omega$$

where, $u \in \Re^2$ is displacement vector field

λ and μ are the Lame constants

Future of sparse structured factorization

- **3D problems**
- **parallel algorithms**
- **Rank analysis for more problems**
- **Nonsymmetric, indefinite problems**

Final remark

- **Sparse factorization algorithms are very difficult to scale up**
  - Numerics, combinatorics, high degree dependency, but modest parallelism is achievable.
- **Still, indispensible tool for difficult problems**
  - As preconditioner, acceleration techniques, can be effectively used to improve numerics for iterative methods.