Improving Scalability of Sparse Direct Linear Solvers

X. Sherry Li

Lawrence Berkeley National Laboratory

CIG Workshop on Geodynamics and Scientific Computing

UT Austin

October 16-17, 2006

Acknowledgement

- Supports from DOE, NSF
- Collaborators:
  - **James Demmel, UC Berkeley**
  - **Laura Grigori, INRIA, France**
  - **Ming Gu, UC Berkeley**
  - **Panayot Vassilevski, Lawrence Livermore National Lab**
  - **Jianlin Xia, UCLA**

Sparse direct linear solver

- Solve A x = b
  - **Example: A of dimension 10^6, only 10 ~ 100 nonzeros per row**
  - **No restriction on sparsity pattern (as opposed to structured matrices)**
- Algorithm: LU factorization: A = LU, followed by lower/upper triangular solutions
  - **Store only nonzeros and perform operations only on nonzeros**
- Distinctions from dense solvers
  - **Need to accommodate fill-in elements**
  - **Reorderings to maintain numerical stability, preserve sparsity, and maximize parallelism: $P_r A P_c^T = L U$**
  - **Irregular, indirect memory access; High communication-to-computation ratio (latency-bound)**

Available codes

- 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, distributed-memory, out-of-core**
- Distributed-memory solvers: usually MPI-based
  – **SuperLU_DIST [Li, Demmel, Grigori]**
    - **Accessible from PETSc, Trilinos**
  – **MUMPS, PasTiX, WSMP, . . .**

SuperLU software status

	SuperLU	SuperLU_MT	SuperLU_DIST
Platform	Serial	SMP	Distributed
Language	C	C + Pthreads (or pragmas)	C + MPI
Data type	Real/complex, Single/double	Real, double	Real/complex, Double

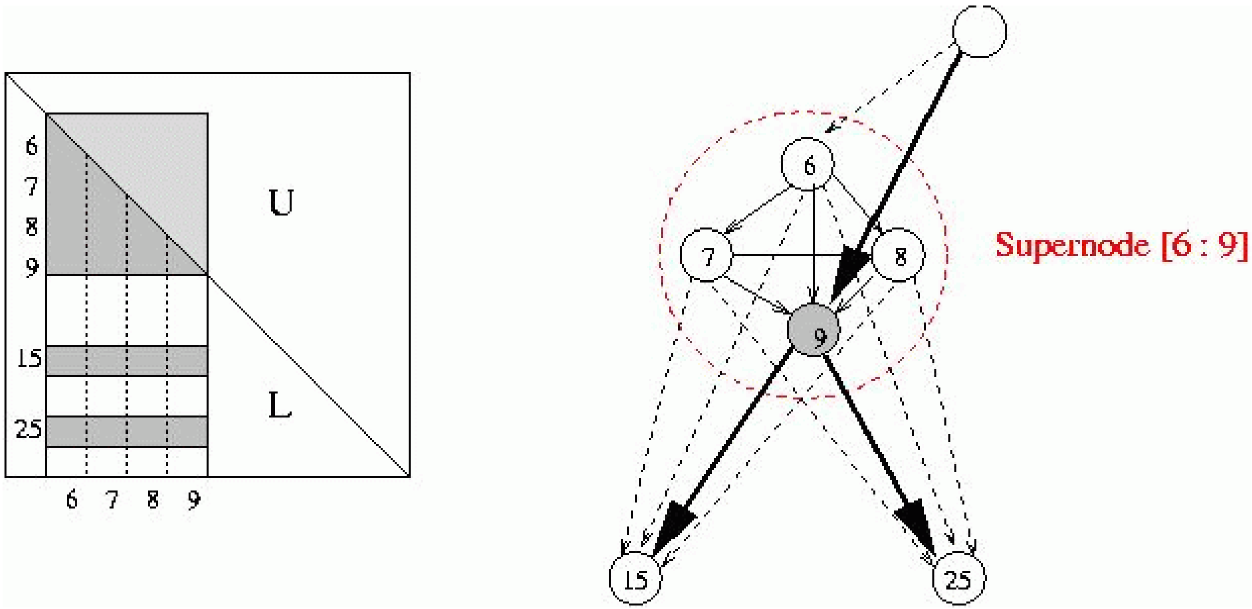
- With Fortran interface
- SuperLU_MT similar to SuperLU both numerically and in usage

SuperLU_DIST major steps:
(parallelization perspectives)

- Static numerical pivoting: improve diagonal dominance
  - **Currently use MC64 (HSL); Parallelization underway [J. Riedy]**
- Sparsity-preserving ordering
  - **Can use ParMeTis**
- Symbolic factorization: determine pattern of {L\U}
  - **Being parallelized**
- Numerics: factorization, triangular solves, iterative refinement (usually dominate total time)
  - **Parallelized a while ago; Need to improve load balance, latency-hiding**

Supernode

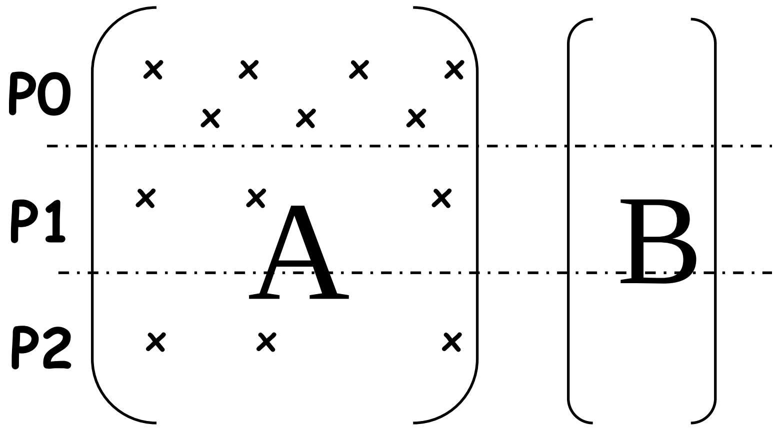
- Exploit dense submatrices in the L & U factors



- Why are they good?
  - **Permit use of Level 3 BLAS**
  - **Reduce inefficient indirect addressing (scatter/gather)**
  - **Reduce symbolic factorization time by traversing a coarser graph**

Distribute the matrices

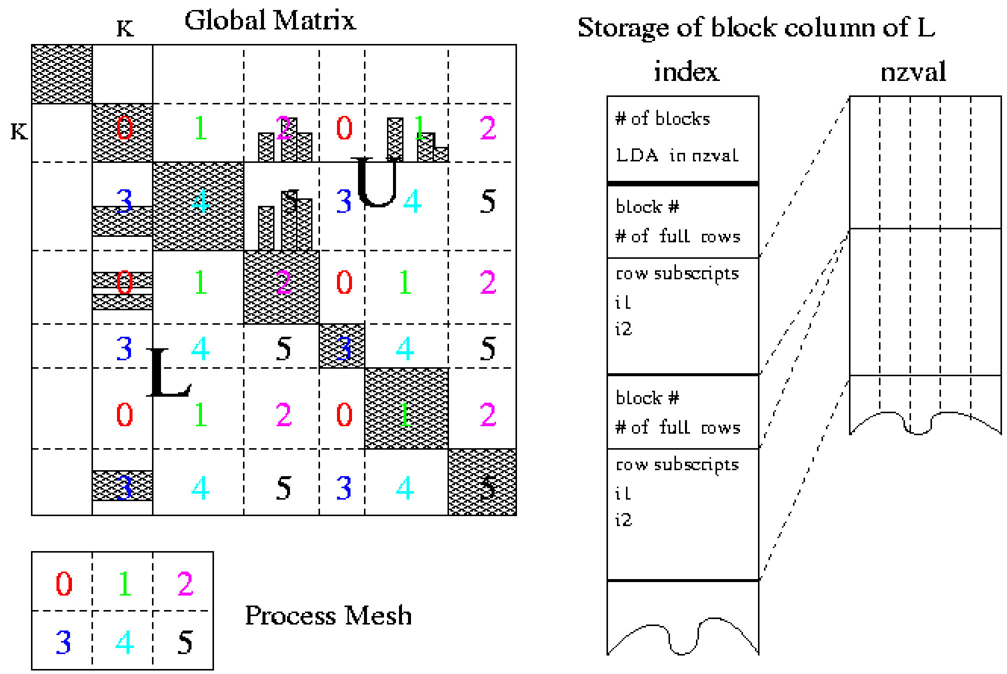
- Matrices involved:
  - **A, B (turned into X) – input, users manipulate them**
  - **L, U – output, users do not need to see them**
- A (sparse) and B (dense) are distributed by block rows



Local A stored in ***Compressed Row Format***

  - **Natural for users, and consistent with other popular packages: e.g. PETSc**

2D block cyclic layout for {L\U}



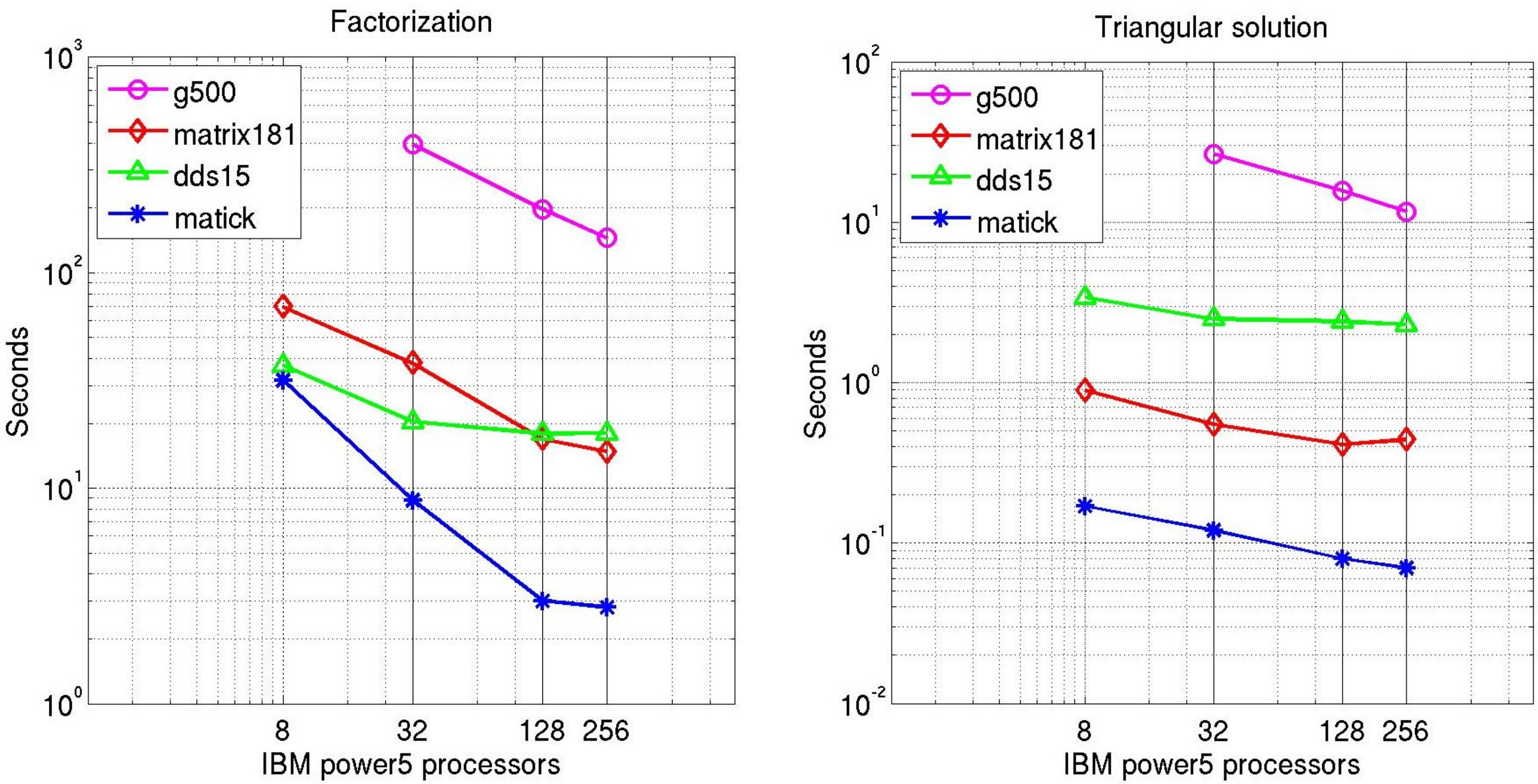
- Good for scalability, load balance
- "Re-distribution" phase to distribute the initial values of A to the 2D block-cyclic data structure of L & U
  - **All-to-all communication, entirely parallel**
  - **< 10% of total time for most matrices**

Examples

Name	Application	Data type	N	|A| / N Sparsity	|L\U| (10^6)	Fill-ratio
g500	Quantum Mechanics (LBL)	Complex	4,235,364	13	3092.6	56.2
matrix181	Fusion, MHD eqns (PPPL)	Real	589,698	161	888.1	9.3
dds15	Accelerator, Shape optimization (SLAC)	Real	834,575	16	526.6	40.2
matick	Circuit sim. MNA method (IBM)	Complex	16,019	4005	64.3	1.0

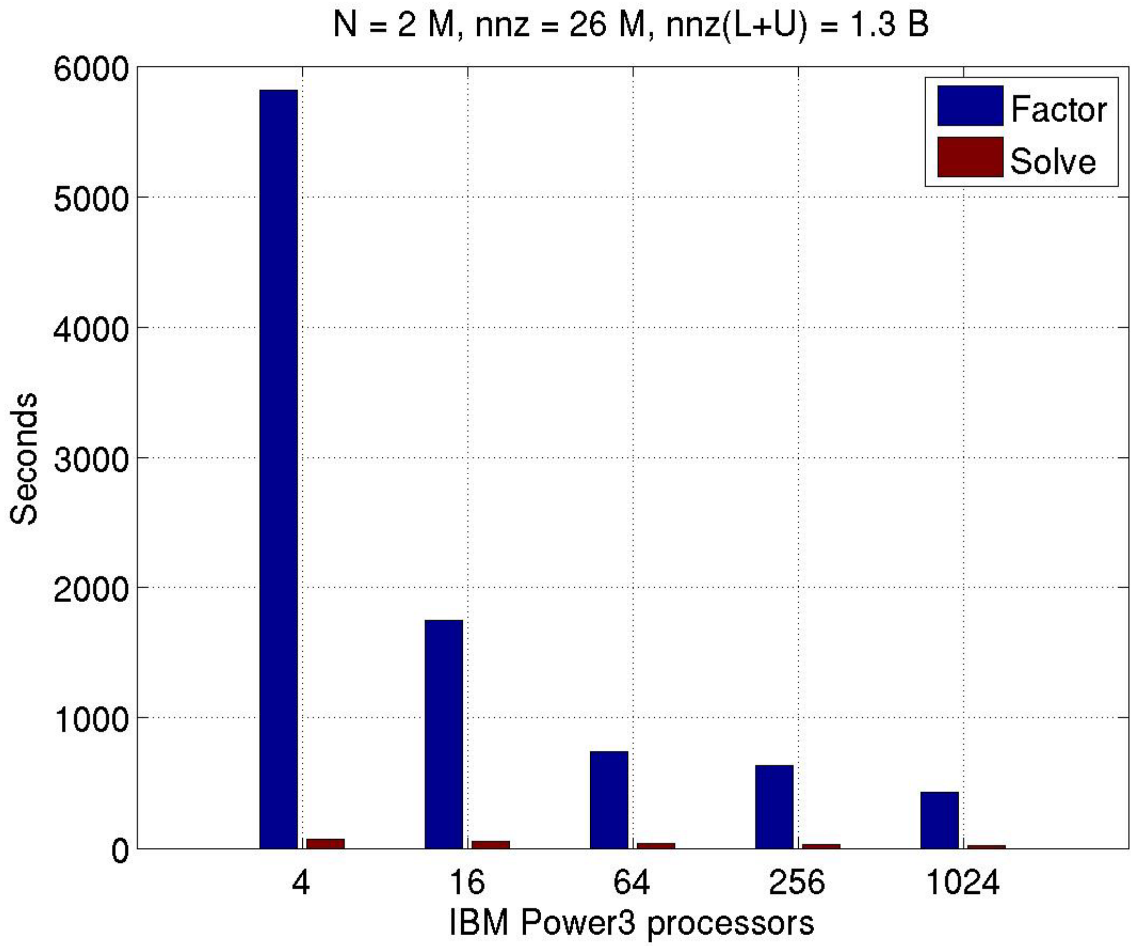
- Sparsity-preserving ordering: MeTis applied to structure of A'+A

Performance on IBM Power5 (1.9 GHz)



- Up to 454 Gflops factorization rate

Performance on IBM Power3 (375 MHz)



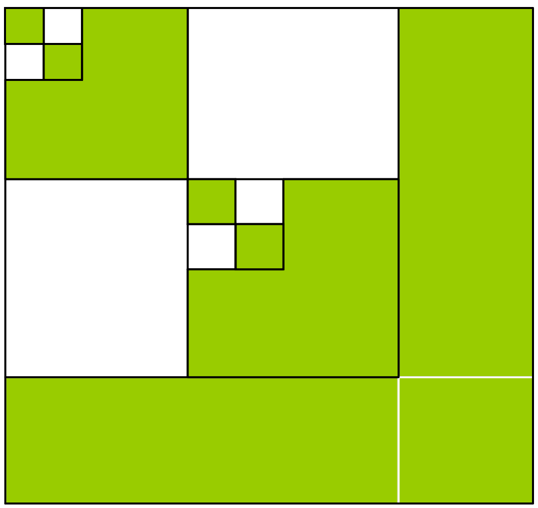
- Quantum mechanics, complex

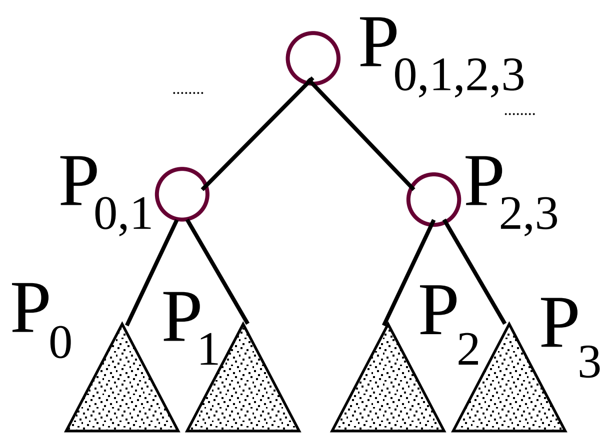
Parallelizing symbolic factorization

- Serial algorithm is fast (usually < 10% total time) but requires entire structure of A, limiting memory scalability
- Parallel approach
  - **Use graph partitioning to reorder/partition matrix.**
    - **ParMetis on structure of A + A'**
  - **Exploit parallelism given by this partition (coarse level) and by a block cyclic distribution (fine level)**
- Summary of results
  - **Memory: up to 25x reduction of symbolic fact.**
    up to 5x reduction of the entire solver
  - **Runtime: up to 14x speedup of symbolic fact.**
    up to 20% faster of the entire solver

Matrix partition

- Separator tree
  - Balanced tree with balanced data distribution
  - Exhibits computational dependencies
    - If node j updates node k, then j belongs to subtree rooted at k.





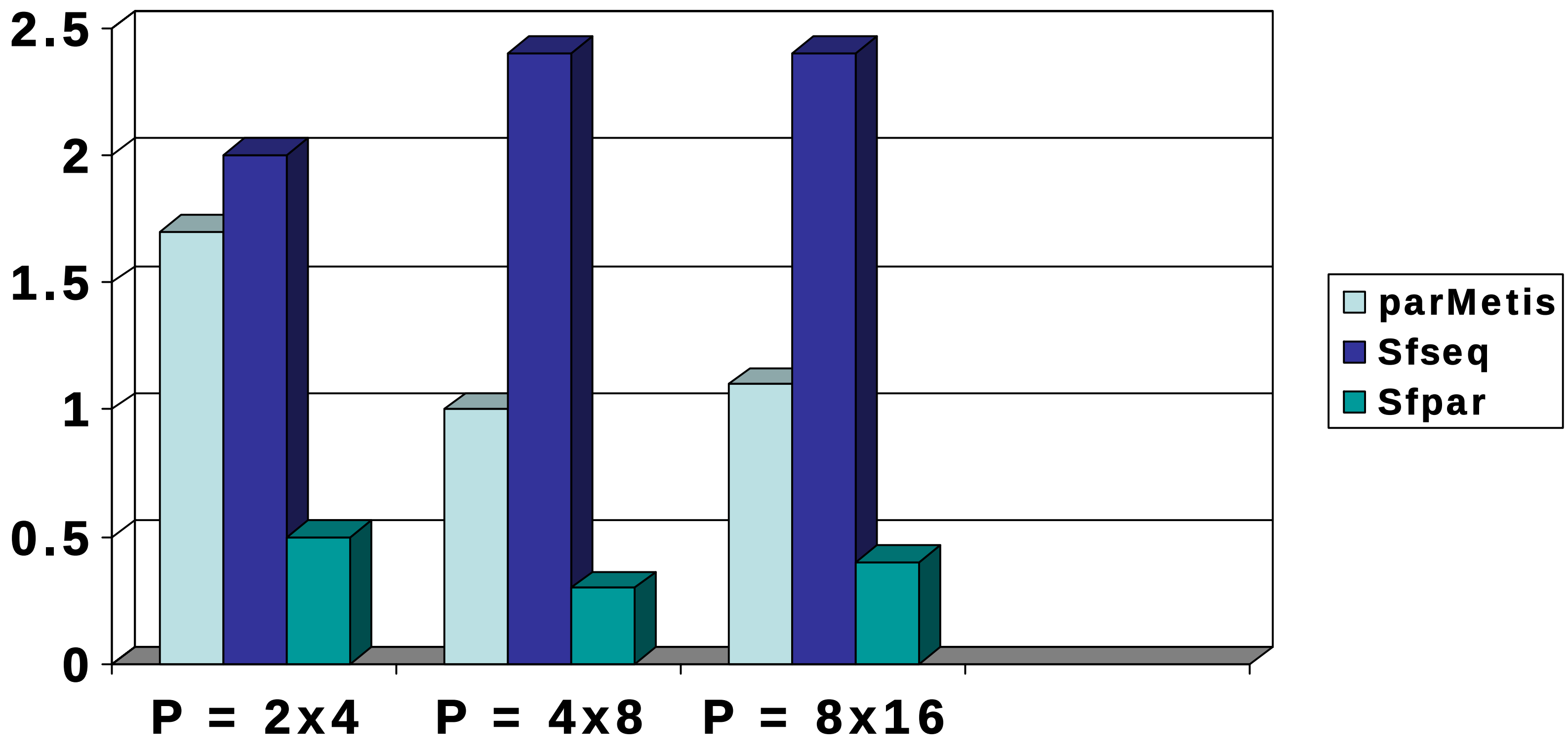
Fluid flow (1/1)

- bbmat: n = 38,744, nnz = 1.8 M, 34 M fill-ins using ParMetis on one processor
- Memory usage:
  - **SFseq (symbolic sequential), SFpar (symbolic parallel)**
  - **Entire solvers: SLU_SFseq, SLU_SFpar**

Memory needs(MB)	*P=8*	*P=32*	*P=128*
Nnz(L+U)*10^6	35.0	36.7	36.6
SFseq	35.6	36.5	40.7
SFpar (max)	6.7	3.0	1.6
SFseq / SFpar	5.3	12.2	25.4
Factor	44.7	13.1	4.0
SLU_SFseq	86.4	52.1	45.3
SLU_SFpar	58.4	19.5	8.0

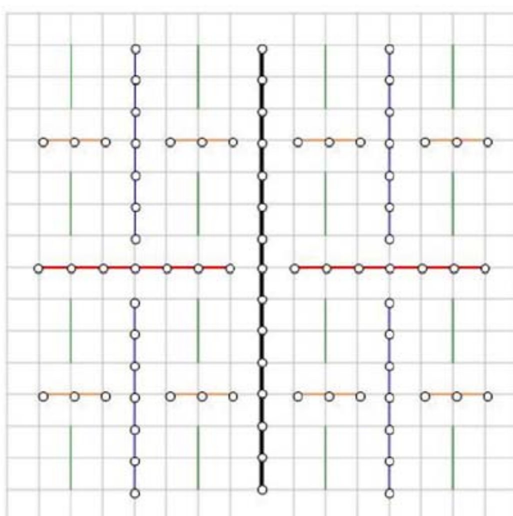
Fluid flow (2/2)

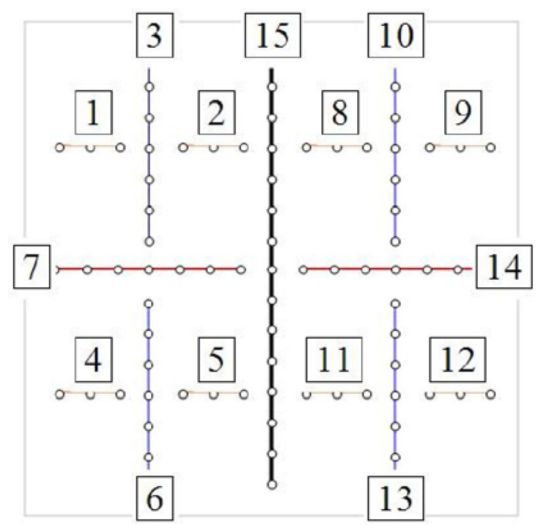
- Runtime in seconds



Fast solver

- In the spirit of fast multipole, but for matrix inversion
- Model problem: discretized system Ax = b from certain PDEs, e.g., 5-point stencil on k x k grid, $n = k^2$
- Nested dissection ordering gave optimal complexity in exact arithmetic [Hoffman/Martin/Ross]
  - **Factorization cost: $O(n^{1.5})$ (3D: $O(n^2)$)**





Exploit low-rank property

- Consider top-level dissection:
- S is full
  - **Needs $O(k^3)$ to find u_3**

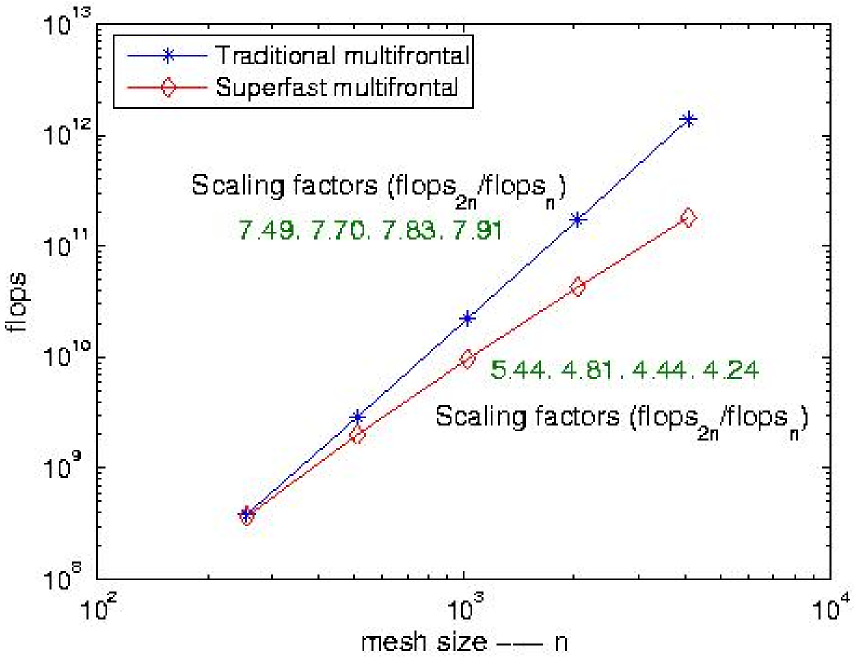
$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}$$

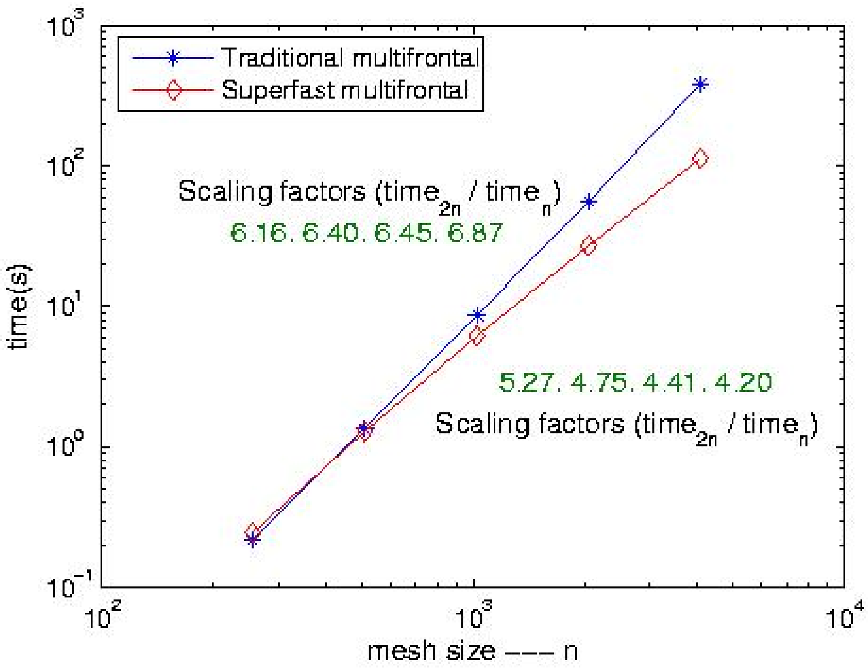
$$S\, u_3 = f_3 - A_{31} A_{11}^{-1} f_1 - A_{32} A_{22}^{-1} f_2$$

- But, off-diagonal blocks of S has low numerical ranks (e.g. 10~15)
  - **u_3 can be computed in $O(k)$ flops**
- Generalize to multilevel dissection: all diagonal blocks corresp. to the separators have the similar low rank structure
- Low rank structures can be represented by hierarchical semi-separable (HSS) matrices [Gu et al.] (… think about SVD)
- Factorization complexity … essentially linear
  - **2D: $O(p\, k^2)$, p is related to the problem and tolerance (i.e., numerical rank)**
  - **3D: $O(c(p)\, k^3)$, c(p) is a polynomial of p**

Results of the model problem

- Flops and runtime comparison





Summary

- Current factorization algorithms can scale to 1000s processors

- New “fast solver” has potential of scaling to tera/petascale; demonstration remains open