Rapid Development of Parallel High-Performance OOC Solvers for Electromagnetics

T. Joffrain[1], J.B. Layton[2], E.S. Quintana-Ortí[3], R.A. van de Geijn[1]

[1]Dept. of Computer Sciences,
The University of Texas at Austin, TX (USA)

[2]Lockheed-Martin Aeronautics Company,
Dept. 6c6m, Marietta, GA (USA)

[3]Depto. de Ingeniería y Ciencia de Computadores
Universidad Jaume I de Castellón (Spain)

PARA'04 - June 2004

Motivation

- Boundary Element Methods (BEM) for the solution of integral equations in electromagnetics and acoustics lead to dense linear systems of $\mathcal{O}(n^5 - n^6)$.
- Sometimes, fast multipole methods are not the solution.

A system with $n = 1{,}000{,}000$ requires $\approx$ 8 Tbytes of memory!

$$\Longrightarrow$$

- Necessary to use disk storage.
- Important to reduce I/O overhead: OOC techniques.

Outline

- LU factorization.
- Slabwise OOC implementation.
- Update of an LU factorization.
- Tiled OOC implementation.
- Numerical experiments.
- Conclusions.

LU factorization

Interest: First stage for solving a linear system $Ax = b$:

1. Factorize $A = LU$, with L/U lower/upper triangular.
2. Solve $Ly = b$.
3. Solve $Ux = y$.

We concentrate in stage 1 as it presents the major difficulties for OOC.

Several implementations are possible: right-looking, left-looking, etc.

Blocked variants are used for performance.

LU factorization (Cont.)

Partial (rowwise) pivoting is added for numerical stability:

- Pivoting (as in LINPACK):

$$L_n^{-1} P_n \cdots L_2^{-1} P_2 L_1^{-1} P_1 A = U.$$

  - Row permutations are applied to A as it is transformed.
  - After the factorization, L is not provided explictly.

- In LAPACK permutations are reorganized:

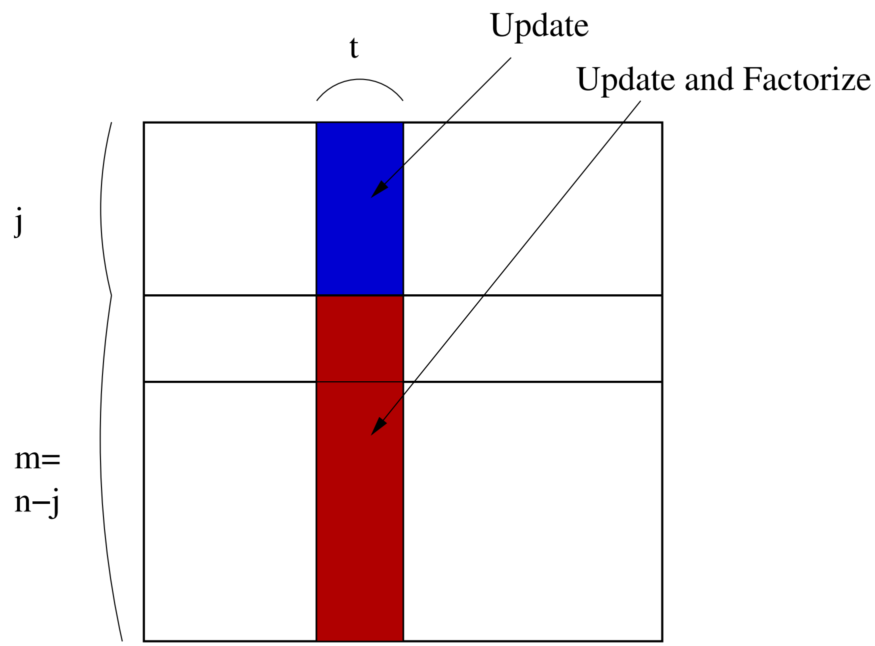
$$L_n^{-1} P_n \cdots L_2^{-1} P_2 L_1^{-1} P_1 A =$$
$$\hat{L}_n^{-1} \cdots \hat{L}_2^{-1} \hat{L}_1^{-1} P_n \cdots P_2 P_1 A = L^{-1} P A = U.$$

  - Row permutations are applied to L and A.

Slabwise (left-looking) OOC implementation

Proceed by bringing slabs (panels) of t columns into memory:

- Left-looking algorithm minimizes memory writes.
- Working with slabs allows to respect partial pivoting.



Slabwise (left-looking) OOC implementation (Cont. I)

Proceeding by slabs of dimension $n \times t$:

- Read the slab and the lower trapezoidal matrix to the left, and write back the slab

$$\rightarrow \; 2n^2 + n^3/3t \text{ I/O operations.}$$

Thus,

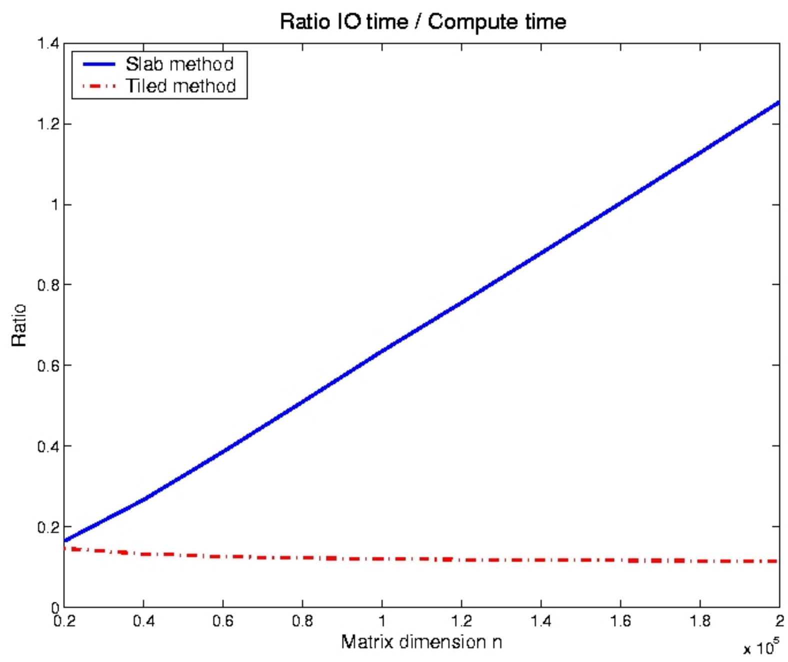
$$\frac{\mathrm{I/O}}{\mathrm{Computation}} = \frac{2n^2 + n^3/3t}{2n^3/3} \approx \frac{1}{2t}.$$

As n grows, $t \downarrow$, so that I/O becomes considerable!

Slabwise (left-looking) OOC implementation (Cont. II)

Tiled approaches (use square tiles of order t) solve this problem.

- System: 1 Gbyte of RAM, I/O rate=40 MBytes/sec., 2.6 Gflops.



However, it does not allow partial pivoting to be used!

Update of an LU factorization

Consider the matrix

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}.$$

Given that $PA = LU$ is available,. . .

how can we use it to obtain a factorization of the whole matrix?

Interest: modeling the radar signature of an airplane (and also OOC; see in a moment).

Solve j=0,1,2,. . .

$$\begin{pmatrix} A & B_j \\ C_j & D_j \end{pmatrix} \begin{pmatrix} x_{1,j} \\ x_{2,j} \end{pmatrix} = \begin{pmatrix} b_{1,j} \\ b_{2,j} \end{pmatrix},$$

with A much bigger than B_j, C_j, and D_j.

Update of an LU factorization (Cont. I)

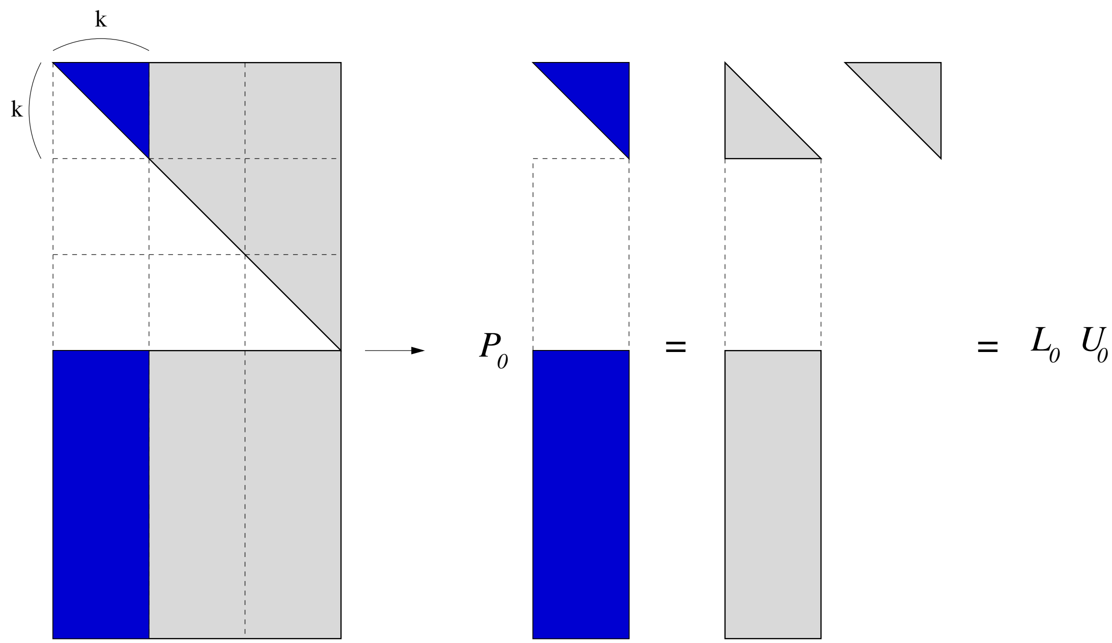
LU factorization of $2t \times 2t$ $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$ with incremental pivoting:

1. Factor $PA = LU$.
2. Update B consistently $B := L^{-1}PB$.
3. Factor $\bar{P}\begin{pmatrix} U \\ C \end{pmatrix} = \begin{pmatrix} \bar{L}_1 \\ \bar{L}_2 \end{pmatrix}\bar{U} = \bar{L}\bar{U}$.
4. Update $\begin{pmatrix} B \\ D \end{pmatrix} := \begin{pmatrix} \bar{L}_1^{-1}B \\ D - \bar{L}_2(\bar{L}_1^{-1}B) \end{pmatrix}$.
5. Factor $\hat{P}D = \hat{L}\hat{U}$.

- Dummy approach: $4t^3/3$ additional flops.
- The key lies in exploiting the structure in stages 3 and 4.

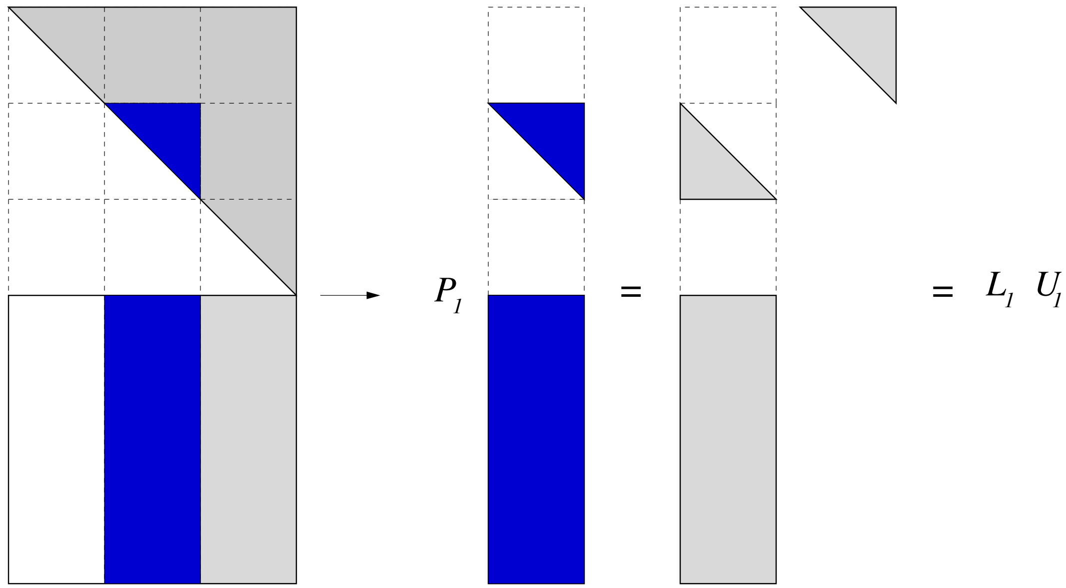
Update of an LU factorization (Cont. II)

Stage 3: Factor $\begin{pmatrix} U \\ C \end{pmatrix}$ proceeding by blocks of $k \times k$.



Application of $L_0^{-1} P_0$ to the remaining blocks preserves the structure.

Update of an LU factorization (Cont. III)

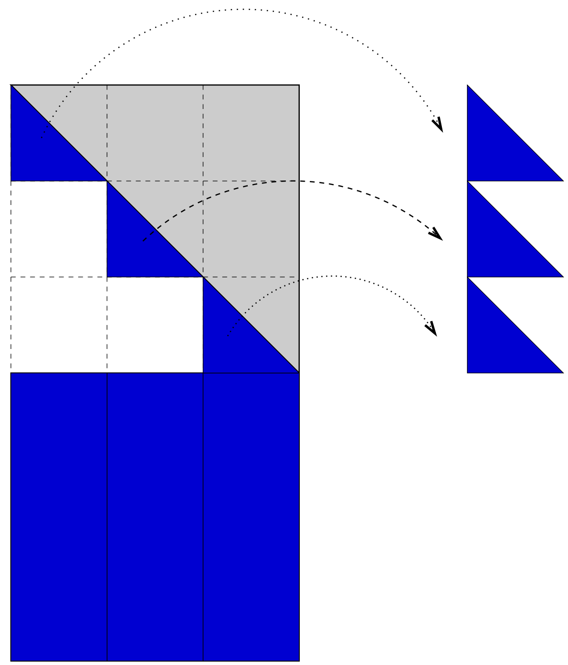


. . . and so on.

Update of an LU factorization (Cont. IV)

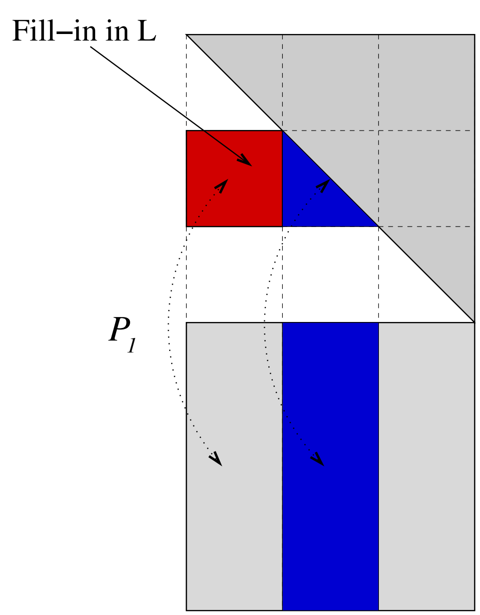
As U already stores the factors of $PA = LU$ from stage 1

$\rightarrow$ Need additional storage for t/k (lower) triangular $k \times k$ factors.



Update of an LU factorization (Cont. V)

Reorganizing $L_2^{-1}P_2L_1^{-1}P_1L_0^{-1}P_0 \begin{pmatrix} U \\ C \end{pmatrix} = \begin{pmatrix} \bar{U} \\ 0 \end{pmatrix}$ as in LAPACK destroys the structure of L increasing the cost of the update in stage 4!



Numerical Stability

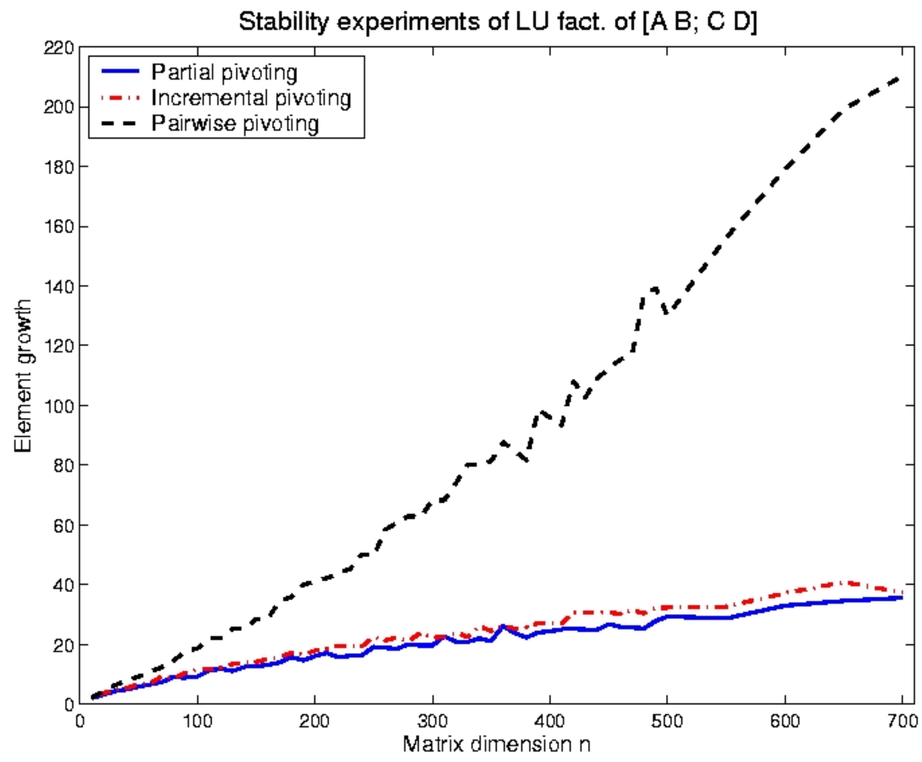
Stability of LU factorization is linked with *element growth*.

Growth	Worst case	Avg. case
Complete pivoting	$\mathcal{O}(n^{1+x})$	$n^{1/2}$
Partial pivoting	2^{n-1}	$n^{2/3}$
Pairwise pivoting	4^{n-1}	n

- Theoretically, both partial and pairwise pivoting are unstable.
- However, practice tells us to trust partial pivoting.
- Pairwise pivoting is not that much worse (Wilkinson, Demmel, Sorensen).
- Incremental pivoting is a combination of partial/pairwise pivoting.
- Solutions can be improved by iterative refinement: unexpensive.

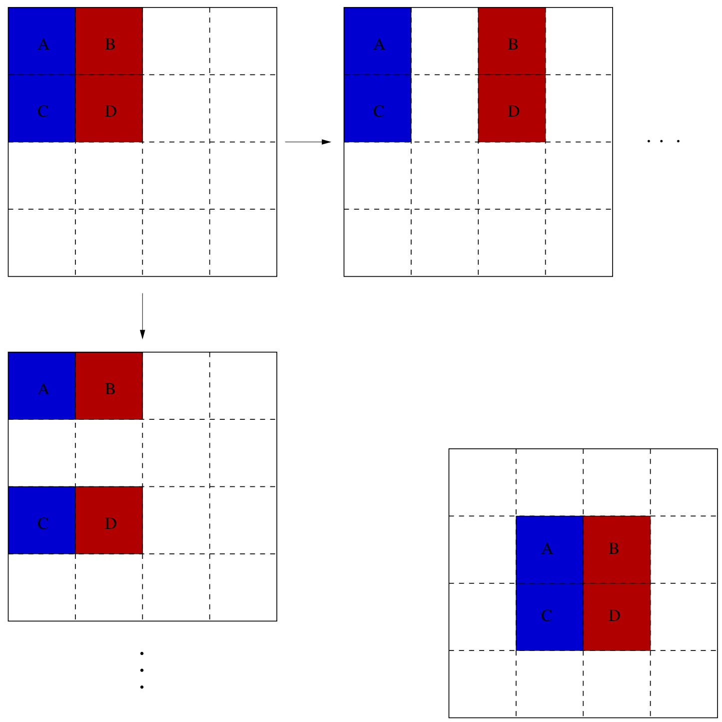
Numerical Stability

Averaged element growth; 10 repetitions for each matrix size.



Tiled OOC implementation

Just employ a generalization of the LU update idea...



Tiled OOC implementation (Cont. I)

- One-tile, two-tile, three-tile algorithms are possible.
- In practice, we use a right-looking three-tile algorithm.

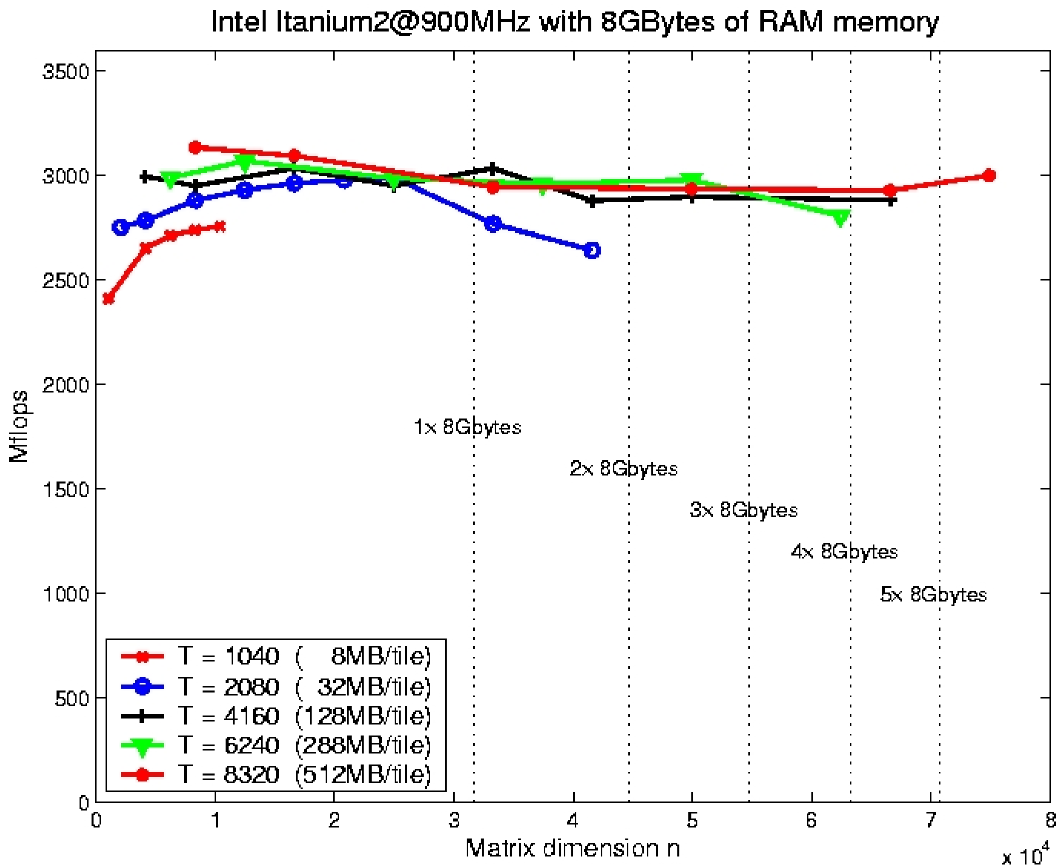
Rapid development? Use of FLAME APIs described in previous talk:

- From paper to running MATLAB code in one morning.
- 463 lines of code using MATLAB, 923 using FLAMEC, 1825 using PLAPACK+POOPLAPACK.

Numerical Experiments

- All results obtained on an Intel Itanium2 (R) architecture (900MHz) with 8 Gbytes of RAM memory (donated by HP).
- Use of IEEE double-precision arithmetic.
- Theoretical peak of 3.6 Gflops.
- In practice, 3.1 Glops for in-core LU factorization ($5{,}200 \times 5{,}200$ matrix).

Numerical Experiments (Cont. I)



Concluding Remarks

- Update of an LU factorization.
- Tiled algorithm for OOC LU factorization $\longrightarrow$ linear systems.
- Easy implementation using FLAME, PLAPACK, and POOCLAPACK.
- Good performance and scalability.
- Numerical stability different from that of partial pivoting
  $\longrightarrow$ iterative refinement.