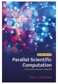
# High-Performance LU Decomposition

## Section 2.5 of Parallel Scientific Computation, 2nd edition

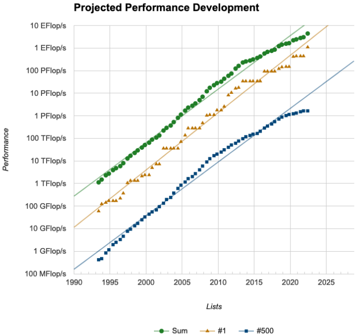
Rob H. Bisseling

Utrecht University



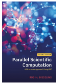


# High-Performance LINPACK (HPL)




Source: TOP500 June 2022 `https://www.top500.org`

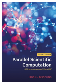
- HPL performs an LU decomposition with partial row pivoting and solves a triangular system, to solve a dense linear system.
- HPL is used to obtain performance results for the TOP500 of supercomputers.



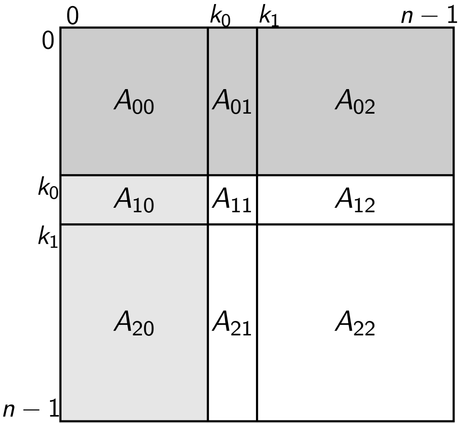
# Selective procrastination for higher performance

- Selective procrastination helps to achieve higher performance by creating bulk, i.e., large batches of work.
- Here, we postpone all updates of the submatrix $A(k_1 : n-1, k_1 : n-1)$ from stages $k$ with $k_0 \leq k < k_1$ of the LU decomposition.
- $b = k_1 - k_0$ is the algorithmic block size
- The main update loop then becomes:

**for** $k := k_0$ **to** $k_1 - 1$ **do**
  **for** $i := k_1$ **to** $n - 1$ **do**
    **for** $j := k_1$ **to** $n - 1$ **do**
      $a_{ij} := a_{ij} - a_{ik} a_{kj}$;

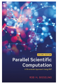


# Submatrices

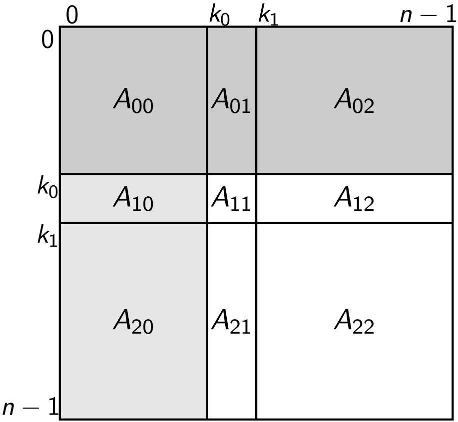



- Submatrices $A_{00}, A_{01}, A_{02}$ are finished at the start of stage $k_0$.
- Operations on $A_{11}, A_{21}$ are carried out immediately.
- Permutation operations on $A_{10}, A_{20}$ and matrix update on $A_{12}, A_{22}$ are delayed and then done in bulk.





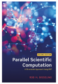
# Tall-and-skinny matrices $A_{21}$ and $A_{12}$




- The matrix update can be formulated as

$$A_{22} := A_{22} - A_{21}A_{12}.$$

- Cost: $2(n-k_1)^2 b$ flops for $(n-k_1)^2$ data, so potentially $b\times$ reuse of data. Cache-friendly!

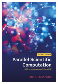


# Choice of algorithmic block size $b$

- Adding the cost for $k_0 = 0, b, \ldots, n - b$ gives:

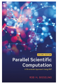
$$T_{\mathrm{blocks}} = \frac{2n^3}{3} - bn^2 + \frac{b^2 n}{3}.$$

- Advantage for small $b$: more flops are performed at an increased computing rate.
- Advantage for large $b$: data reuse is higher, so the rate increase is larger.
- Empirical approach: start with $b = 1$ and double its value until the flop rate saturates, usually in the range $b =$ 32–256.
- This is beneficial for both sequential and parallel LU decomposition.



# Procrastination also reduces communication cost

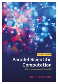
- Delaying the row swaps creates more bulk: $2b$ rows move at the same time, instead of 2 rows.
- For $b \geq \sqrt{p}/2$, all processors are now expected to be involved in the swaps, instead of only 2 processor rows.
- This reduces the cost of the swaps by a factor of $\frac{\sqrt{p}}{2}$ from $\frac{n^2}{\sqrt{p}}g$ to $\frac{2n^2}{p}g$.
- The extra synchronization cost incurred is $\frac{2n}{b}l$.



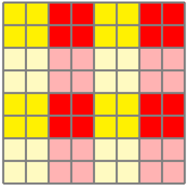
# Avoiding global synchronization

- To reduce the cost of an algorithm, we want to avoid everything: computation, communication, and synchronization.
- Minimizing and balancing flops avoids computation.
- Methods like two-phase broadcasting or 3D matrix multiplication (Exercise 2.6) avoid communication.
- BSP is based on global synchronizations. We present methods to avoid them, such as combining supersteps.
- Usually there is a trade-off between these three objectives and also the amount of memory available.





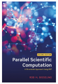
# Block-cyclic distribution



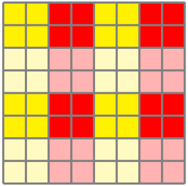
$\beta = 2,\ M = N = 2,\ n = 8$

- The block-cyclic distribution with block size $\beta$ assigns matrix elements to processors by

$$\phi_0(i) = (i \text{ div } \beta) \bmod M, \quad \text{for } 0 \le i < n.$$
$$\phi_1(j) = (j \text{ div } \beta) \bmod N, \quad \text{for } 0 \le j < n.$$

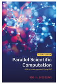


# Using the block-cyclic distribution



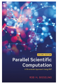
$\beta = 2,\ M = N = 2,\ n = 8$

- The block-cyclic distribution reduces the synchronization cost $\mathcal{O}(nl)$ of LU decomposition without pivoting by a factor $\beta$.
- The load imbalance $\mathcal{O}(\frac{n^2}{\sqrt{p}})$, however, increases by a factor $\beta^2$.
- The communication cost does not change.
- The extreme case $\beta = 1$ is best for large $n$, because imbalance is of a higher order than synchronization cost. Indexing is also simpler.



# HPL uses the block-cyclic distribution

- High-Performance LINPACK (HPL) by Dongarra, Luszczek, and Petitet (2003) uses the block-cyclic distribution with $\beta = b$, requiring unnecessary compromises.
- HPL performs partial row pivoting, which causes $\mathcal{O}(nl)$ synchronization cost because $n$ pivots are searched for, with a synchronization between the separate searches.
- A better choice is $\beta = 1$:
  - less imbalance;
  - simpler indexing (only one level).



# Summary

- High performance can be attained for LU decomposition by delaying the bulk of the matrix update.
- The delayed work is carried out as the multiplication of two tall-and-skinny matrices, using the BLAS operation DGEMM.
- The algorithmic block size $b$ and the distribution block size $\beta$ need not be the same.
- It is best to use the square cyclic distribution and not the block-cyclic distribution.