Universität Stuttgart
Institut für parallele und verteilte Systeme
Prof. Dr. rer. nat. Miriam Mehl
Dipl.-Ing. Florian Lindner

SS 2016

Parallel Numerics

Exercise 2: Gaussian Elimination

1 Gaussian Elimination

To calculate the solution of a linear equation system

$$Ax = b$$

with a non-singular matrix $A = (a_{i,j}) \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$ the Gauss Elimination algorithm can be applied. The algorithm decomposes the matrix A in a lower triangular matrix L and an upper triangular matrix U: $A = L\,U$ (cp. lecture notes).

A procedure for calculating the upper triangular matrix $U =: A^{(n)}$ from the matrix A can be given as follows:

i) Define $A^{(0)} = (a_{i,j}^{(0)}) := A$

ii) Calculate for $k = 1, \ldots, n-1$ the values $l_{i,k}$, $i = k+1, \ldots, n$ and the matrices $A^{(k+1)} = (a_{i,j}^{(k+1)})$ with

$$l_{i,k} := a_{i,k}^{(k)} / a_{k,k}^{(k)}$$

$$a_{i,j}^{(k+1)} := \begin{cases} a_{i,j}^{(k)} - l_{i,k} a_{k,j}^{(k)} & \text{for } i \in \{k+1, \ldots, n\},\ j \in \{k, \ldots, n\} \\ a_{i,j}^{(k)} & \text{otherwise} \end{cases}$$

An implementation of this algorithm could have the following form ("kij-loop-arrangement"):

```
1 for k = 1 to n - 1
2   for i = k + 1 to n
3     l_{i,k} := a_{i,k} / a_{k,k}
4     for j = k to n
5       a_{i,j} := a_{i,j} - l_{i,k} a_{k,j}
```

2 A Simple Example

i) Given is

$$A = \begin{pmatrix} 2 & 2 & 1 \\ 4 & 2 & 3 \\ 2 & 2 & 2 \end{pmatrix}$$

Compute the Gaussian Elimination $A = L \cdot U$ with the algorithm given above.

Performing the two row operations $A_{2,\bullet} = A_{2,\bullet} - 2A_{1,\bullet}$ and $A_{3,:} = A_{3,\bullet} - 1 \cdot A_{1,\bullet}$ yields the solution:

$$A = L \cdot U = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

ii) Compute the Gaussian Elimination using a pivot search (move the entry with the largest absolute value to the pivot position).

First performing row permutations on A:

$$A = \begin{pmatrix} 4 & 2 & 3 \\ 2 & 2 & 1 \\ 2 & 2 & 2 \end{pmatrix}.$$

Via row operations $A_{2,\bullet} = A_{2,\bullet} - \frac{1}{2}A_{1,\bullet}$, $A_{3,\bullet} = A_{3,\bullet} - \frac{1}{2}A_{1,\bullet}$, and $A_{3,\bullet} = A_{3,\bullet} - A_{2,\bullet}$ we receive:

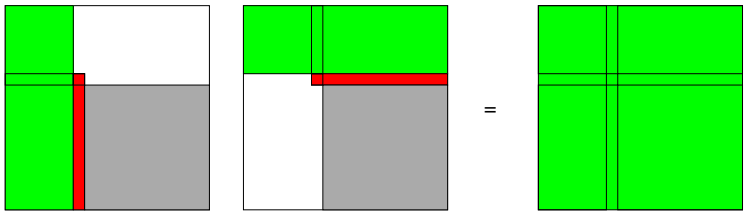
$$A = L \cdot U = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 & 2 & 3 \\ 0 & 1 & -\frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

iii) Why is pivot search needed? What is the additional computational cost?

A pivot search is needed if $A_{k,k} = 0$ at the k-th step of Gaussian elimimation, because it would cause a division by zero. In numerics, even an element close to zero can introduce rounding errors and justifies pivot search. To keep rounding errors small and stabilize the algorithm one usually picks the largest element as pivot. Computational cost is negligible if the row swapping is only saved to an index vector.

3 Parallel Gaussian Elimination

In the lecture, we have defined three steps of the calculation of $L_{2,2}$, $U_{2,2}$, $U_{2,3}$, and $L_{3,2}$ (blocks marked red in the illustration) in the block-wise LU-decomposition



$$\begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} =$$
$$= \begin{pmatrix} L_{11}U_{11} & L_{11}U_{12} & L_{11}U_{13} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} & L_{21}U_{13} + L_{22}U_{23} \\ L_{31}U_{11} & L_{31}U_{12} + L_{32}U_{22} & * \end{pmatrix}$$

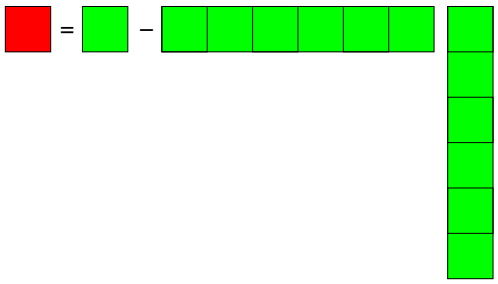
with $A \in \mathbb{R}^{N\times N}, L_{22}, U_{22} \in \mathbb{R}^{m\times m}$, $L_{11}, U_{11} \in \mathbb{R}^{km\times km}$. I.e., we are executing the $k+1$st step of the block-wise elimination algorithm, k "stripes" of width m have already been eliminated.

Develop a distributed memory parallel algorithm including computational and communication steps for all substeps of the block-wise elimination step using

- decompositions of all matrices into $m \times m$ blocks,
- BLAS level-3 routines and small LU-decompositions for operations involving these sub-blocks,
- the number of processors and the maximal amount of storage per process prescribed at the end of the substep descriptions below.

i) Update involved blocks of A using $A \to A - LU$:

$$A_{22} = A_{22} - L_{21}U_{12}$$



Use k processes and up to three small $m \times m$ matrices per processes plus one additional $m \times m$ matrices in the master processes for your parallel algorithm.
We decompose $L_{21} \in \mathbb{R}^{m\times km}$ and $U_{12} \in \mathbb{R}^{km\times m}$ into k blocks L_{21}^j and U_{12}^j of size $(m \times m)$ and store each pair at one processor. A_{22} is stored at the master process.

With this, we

a) compute simultaneously on each of the k processors

$$B_{22}^j = L_{21}^j U_{12}^j,$$

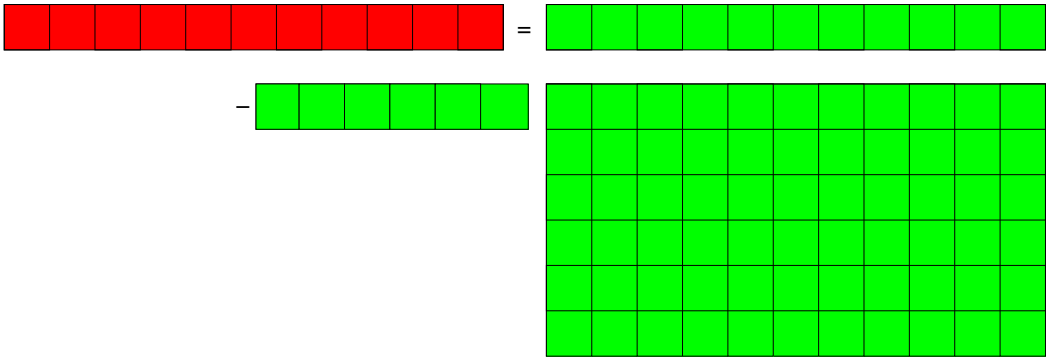
b) communicate B_{22}^j to a master and sum up all B_{22}^j to B_{22} (reduction),

c) compute

$$A_{22} = A_{22} - B_{22}$$

at the master process.

$$A_{23} = A_{23} - L_{21}U_{13}$$



Use $N/m - k$ processes and storage for up to $k + 2$ small $m \times m$ matrices per processes plus a minimal number of additional $m \times m$ matrices in the master processes for your parallel algorithm.
We decompose L_{21} into k blocks $L_{21}^j \in \mathbb{R}^{m\times m}$, U_{13} into $k \cdot (N/m - k)$ blocks $U_{13}^{ji} \in \mathbb{R}^{m\times m}$, and A_{23} into $N/m - k$ blocks A_{23}^i $(j = 1, \ldots, k, i = 1, \ldots N/m - k)$ as sketched in the illustration above.

Use $N/m - k$ processors and store A_{23}^i and $U_{13}^{ji}, j = 1, \ldots, k$ at processor i.

Store L_{21}^j at processor $i, i + k, \ldots, lj + k$ with l chosen such that $N/m - 2k < lj + k \leq N/m - k$. If $N/m - k < k$, store L_{21}^j for $j = N/m - k + 1, \ldots, k$ at the master process (process 1).

With this, we

a) compute simultaneously on each of the $N/m - k$ processors

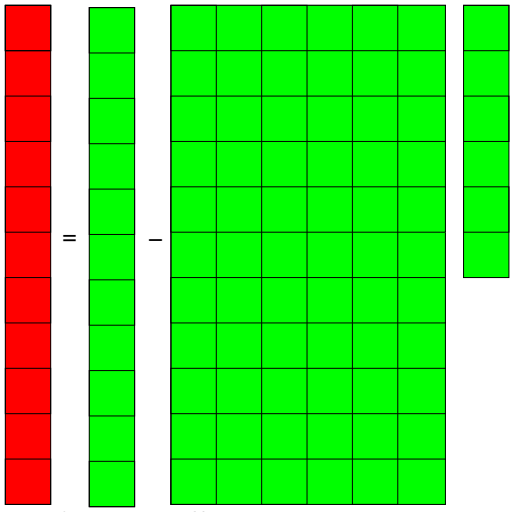
$$A_{23}^i = A_{23}^i - L_{21}^{j(i)} U_{13}^{j(i)i},$$

(if $k > N/m - k$, the master holds more than one block of L_{21} and we choose $j(master) = 1$ as initial value),

b) communicate $L_{21}^{j(i)}$ to process $(i+1) \mod N/m-k$ and set $j(i) = (j(i)-1) \mod k$,

c) continue with a) until all k blocks of L_{21} have been done.

$A_{32} - L_{31}U_{12}$



Use $N/m - k$ processes and storage for up to $k+2$ small $m \times m$ matrices per processes plus a minimal number of additional $m \times m$ matrix in the master processes for your parallel algorithm.
We decompose U_{12} into k blocks $U_{12}^j \in \mathbb{R}^{m\times m}$, L_{13} into $k \cdot (N/m-k)$ blocks $L_{13}^{ji} \in \mathbb{R}^{m\times m}$, and A_{32} into $N/m - k$ blocks A_{32}^i $(j = 1, \ldots, k, i = 1, \ldots N/m - k)$ as sketched in the illustration above.

Use $N/m - k$ processors and store A_{32}^i and $L_{13}^{ij}, j = 1, \ldots, k$ at processor i.

Store U_{12}^j at processor $i, i + k, \ldots, lj + k$ with l chosen such that $N/m - 2k < lj + k \leq N/m - k$.

If $N/m - k < k$, store U_{12}^j for $j = N/m - k + 1, \ldots, k$ at the master process (process 1).

With this, we

a) compute simultaneously on each of the $N/m - k$ processors

$$A_{32}^i = A_{32}^i - L_{31}^{ij(i)} U_{12}^{j(i)},$$

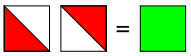
(if $k > N/m - k$, the master holds more than one block of U_{12} and we choose $j(master) = 1$ as initial value),

b) communicate $U_{12}^{j(i)}$ to process $(i+1) \mod N/m-k$ and set $j(i) = (j(i)-1) \mod k$,

c) continue with a) until all k blocks of U_{12} have been done.

ii) Compute the small block-LU-decomposition:

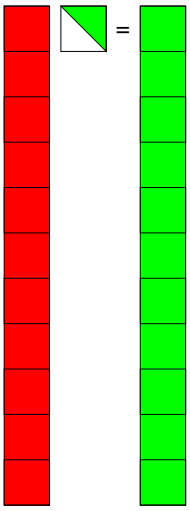
$L_{22}U_{22} = A_{22}$



Given the restriction to matric operations with $m \times m$ blocks, this step can not be further parallelized.

iii) Solve a collection of small independent triangular systems:

$L_{32}U_{22} = A_{32}$



Use $N/m - k$ processes and storage for up to three small $m \times m$ matrices per processes for your parallel algorithm.
Klar, oder?

$L_{22}U_{23} = A_{23}$

Use $N/m - k$ processes and storage for up to three small $m \times m$ matrices per processes for your parallel algorithm.
Klar, oder?