# Band algorithms for matrix multiplication

In these algorithms, matrices are divided into continuous sequences of rows or columns (*bands*). In the simplest case, a band can be a separate row or column.

In the algorithms discussed below, each process is used to compute one row of the resulting matrix product *AB*. In this case, the process must have access to the corresponding row of matrix *A and* the entire matrix *B.* Since simultaneous storage of the entire matrix *B* in all processes of a parallel application requires excessive memory consumption, calculations are organized in such a way that at any given time the processes contain only part of the elements of matrix *B* (one column or one row), and access to the rest is provided using message passing.

When describing band algorithms, it is assumed that the number of processes *N* coincides with the order of the multiplied matrices *A* and *B,* and the matrices are square. In the case where the order of the matrices *M* is a multiple of the number of processes *N, it is sufficient to process bands* containing *M* / *N* rows or columns in each process.

## *Band algorithm 1 (horizontal bands)*

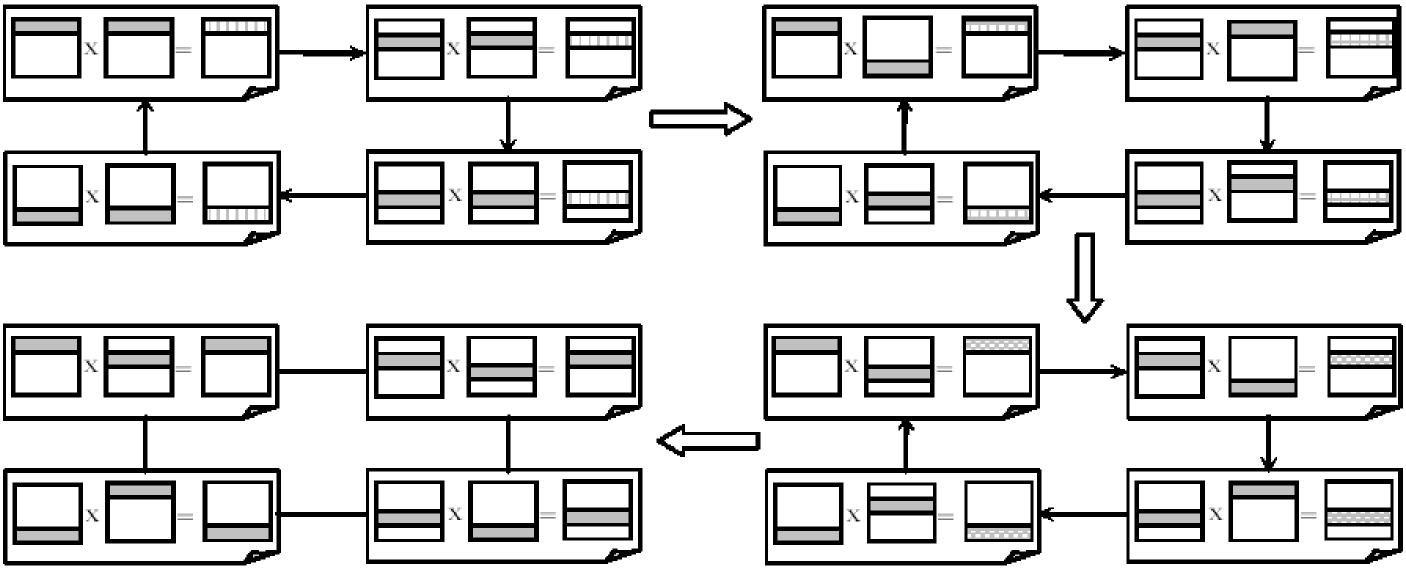
At the beginning, the elements of *the K*-th row of the matrix *A* and the elements of *the K*-th row of the matrix *B* are sent to the process of rank *K.* The elements of the row *c*, which will contain the result, i.e. the corresponding row of the product *AB*, are set to zero.

Then a loop is started (the number of iterations is *N*), during which two actions are performed:

1) *A* and matrix *B* with the same numbers are multiplied, and the results are added to the corresponding row element *c;*
2) the rows of matrix *B* are cyclically sent to neighboring processes (the direction of transfer can be arbitrary: either in increasing or decreasing ranks of processes).

After the loop completes, each process will contain the corresponding row of the product *AB.* All that remains is to send these rows to the master process.

The figure shows a diagram of algorithm 1, provided that cyclic transfer of rows of matrix *B* is performed in the direction of increasing process ranks (process of rank 0 sends its row to process of rank 1, process of rank 1 to process 2, etc.).



## *Band algorithm 2 (horizontal and vertical bands)*

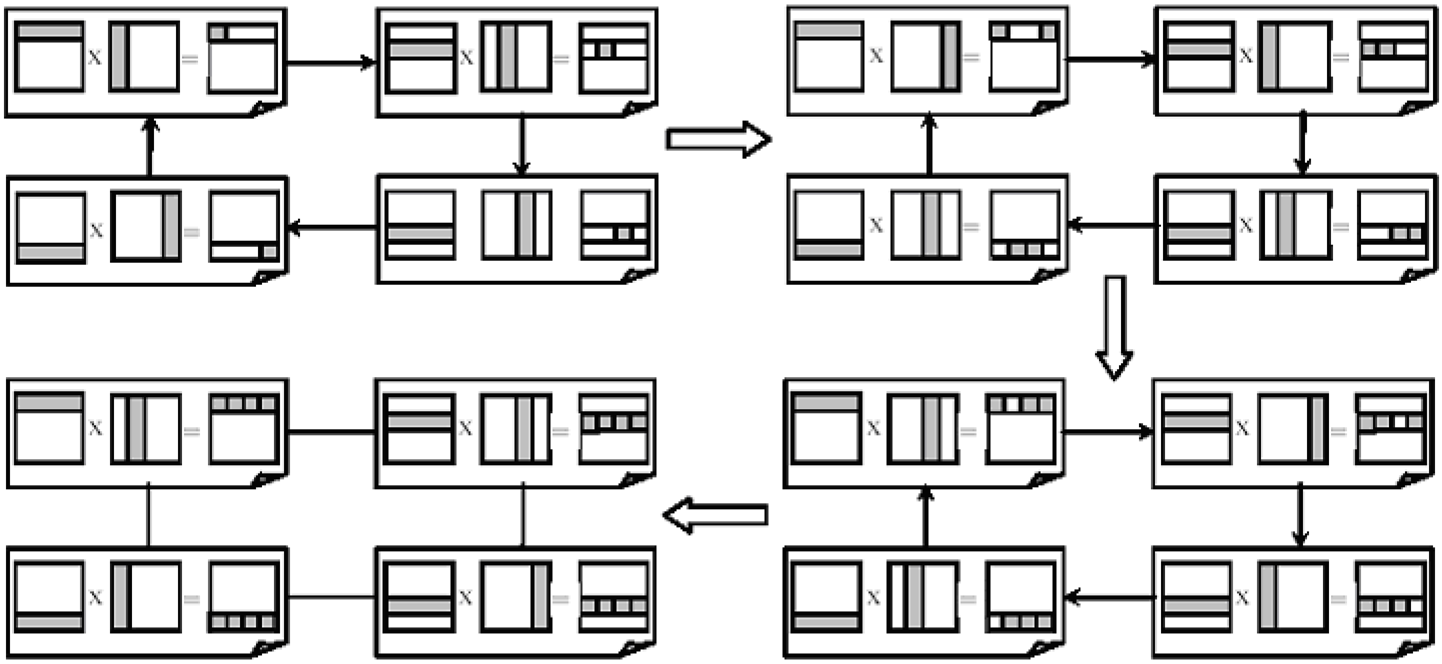
At the beginning, the elements of *the K*-th row of matrix *A* and the elements of *the K*-th column of matrix *B* are sent to the process of rank *K.*

Then a loop is started (the number of iterations is *N*), during which two actions are performed:

1) a multiplication of a row of matrix *A* and a column of matrix *B* contained in this process is performed, and the result is written to the corresponding element of row *c;*
2) the rows of matrix *B* are cyclically sent to neighboring processes (the direction of transfer can be arbitrary: either in increasing or decreasing ranks of processes).

After the loop completes, each process will contain a row *c* equal to one of the strings of the product *AB*. All that remains is to send the rows *c* to the master process.

The figure shows a diagram of algorithm 1, provided that cyclic forwarding of the columns of matrix *B* is performed in the direction of increasing process ranks (process of rank 0 sends its row to process of rank 1, process of rank 1 to process 2, etc.).



# Block algorithms multiplication matrices

In these algorithms, matrices break up on blocks representing submatrices of the original matrices. For simplicity, we will assume that all matrices are square of size $N \times N$, and the number of blocks horizontally and vertically is the same and equal to $q$ (the size of all blocks is equal to $K \times K$, where $K = N / q$). In this case, the matrix multiplication operation can be represented in block form:

$$\begin{pmatrix} A_{00} A_{01} \ldots A_{0q-1} \\ \cdots \\ A_{q-10} A_{q-11} \ldots A_{q-1q-1} \end{pmatrix} \times \begin{pmatrix} B_{00} B_{01} \ldots B_{0q-1} \\ \cdots \\ B_{q-10} B_{q-11} \ldots B_{q-1q-1} \end{pmatrix} = \begin{pmatrix} C_{00} C_{01} \ldots C_{0q-1} \\ \cdots \\ c_{q-10} C_{q-11} \ldots C_{q-1q-1} \end{pmatrix}$$

In this case, each block $C_{ij}$ of matrix *C* is defined as the product of the corresponding blocks of matrices *A* and *B* :

$$C_{ij} = \sum_{s=0}^{q-1} A_{is} B_{sj}$$

When partitioning data blockwise, it is natural to associate with each process the task of calculating one of the blocks of the resulting matrix *C*. In this case, the process must have access to all elements of the corresponding rows of matrix *A* and columns of matrix *B*. Since placing all the required data in each process will lead to their duplication and a significant increase in the amount of memory used, it is necessary to organize calculations in such a way that at each time the processes contain only one block of matrices *A* and *B* required for calculations, and access to the remaining blocks would be provided by message passing.

In these algorithms, it is convenient to introduce a two-dimensional Cartesian topology for processes by associating each process with its coordinates $(i, j)$ in this topology $(i, j = 0, \ldots, q-1)$. It is assumed that the number of processes is equal to $q^{2.}$

## *Fox's block algorithm*

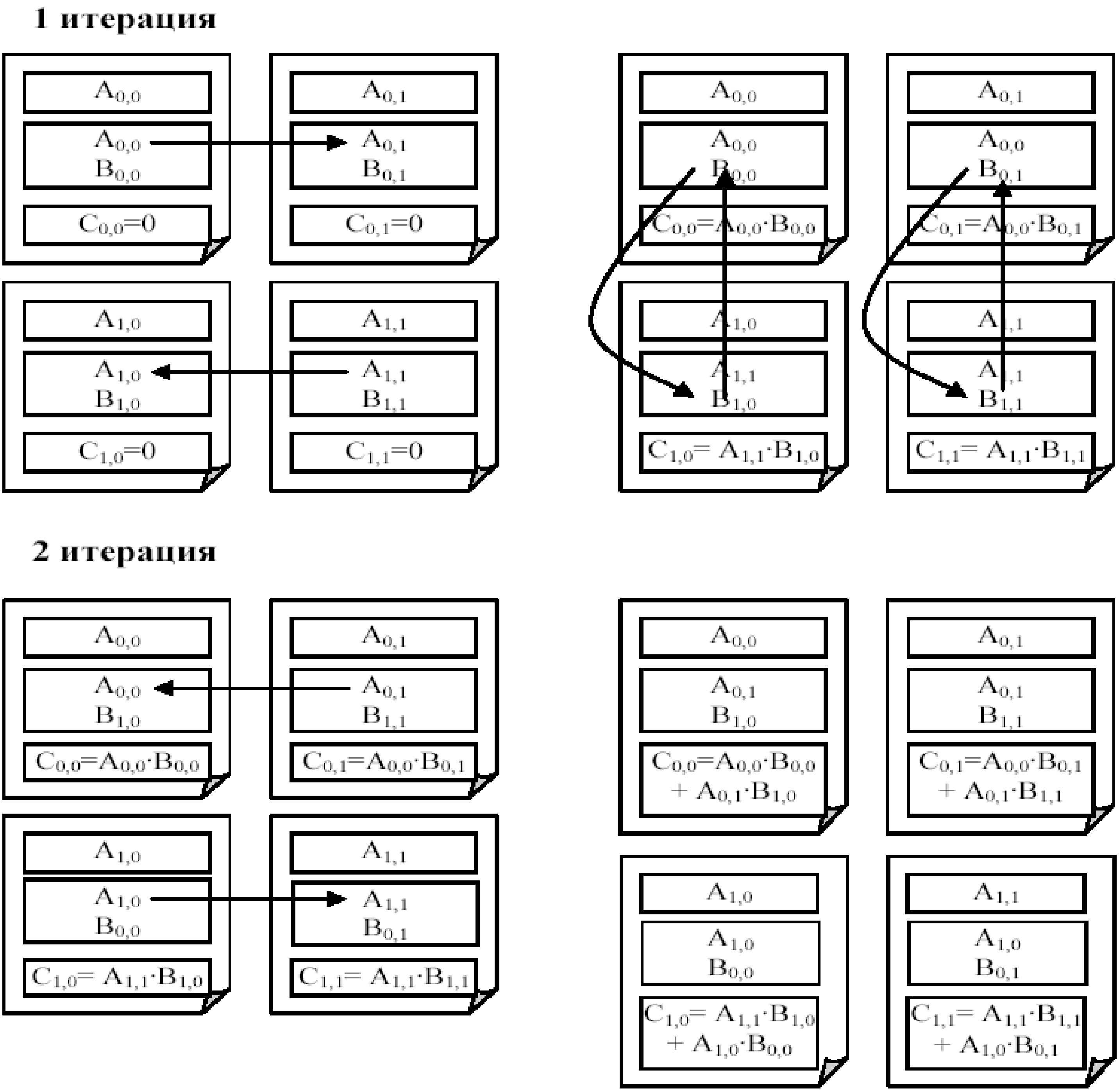
Blocks $A_{ij},$ and $B_{ij}$ of the original matrices are sent to the process with coordinates $(i, j)$. In addition, the matrix $C_{ij},$ intended to store the corresponding block of the resulting product *AB*, is reset to zero.

Then a loop is started in $m$ ($m = 0, \ldots, q - 1$), during which three actions are performed:

1) for each row $i$ ($i = 0, \ldots, q - 1$), block $A_{ij}$ of one of the processes is sent to all processes of the same row; in this case, the index $j$ of the forwarded block is determined by the formula $j = (i + m) \bmod q$;
2) the block of matrix $A$ obtained as a result of such transfer and the block of matrix $B$ contained in the process $(i, j)$ are multiplied, and the result is added to the matrix $C_{ij}$;
3) for each column $j$ ($j = 0, \ldots, q - 1$), cyclic transfer of blocks of matrix $B$, contained in each process $(i, j)$ of this column, is carried out in the direction of decreasing row numbers.

After completion of the loop, each process will contain a matrix $C_{ij}$ equal to the corresponding block of the product $AB$. All that remains is to send these blocks to the master process.

The figure shows a diagram of the Fox's algorithm in the case of $q = 2$.

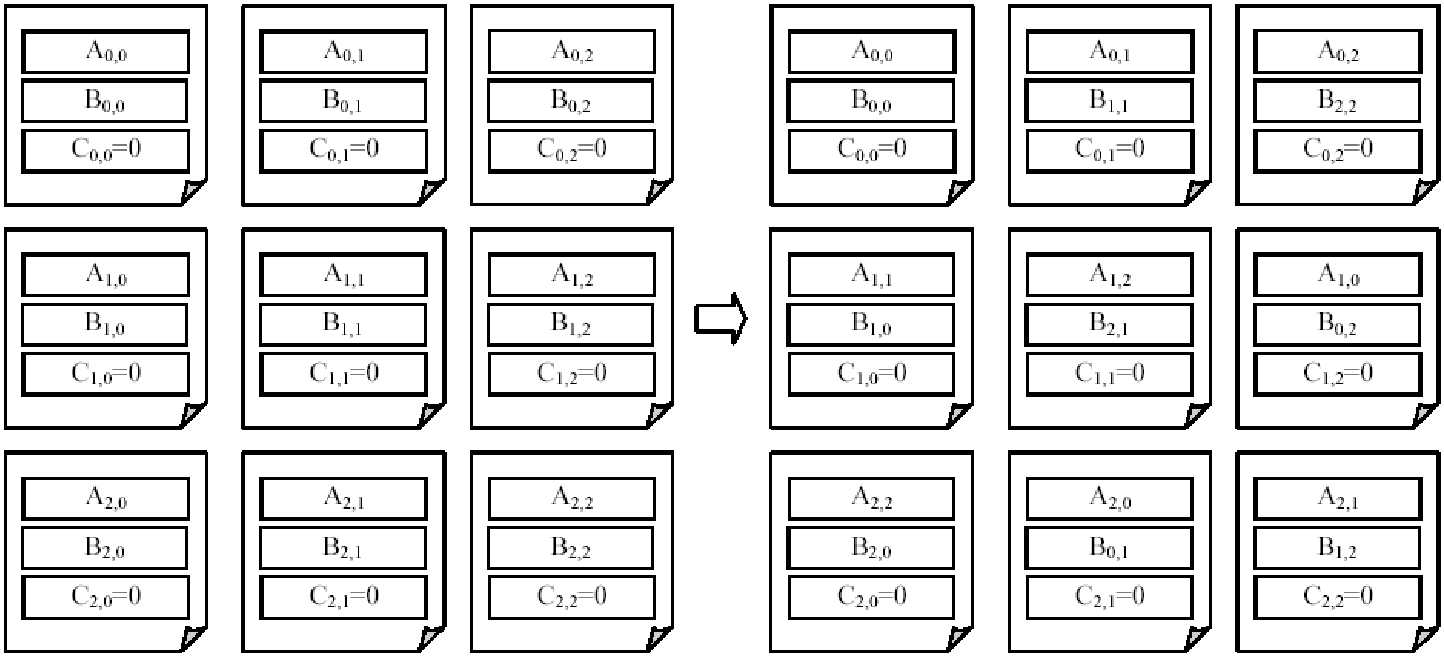


## *Cannon's block algorithm*

Cannon's algorithm differs from Fox's algorithm in two ways. First, the initial transfer of blocks of matrices $A$ and $B$ to processes is performed in such a way that the resulting blocks can immediately be multiplied without any data transfers. Secondly, when organizing a loop, cyclic transfer is carried out not only of blocks of matrix $B$ (by columns), but also by blocks of matrix $A$ (by rows). Initial forwarding actions consist of the following steps:

1) blocks $A_{ij}$ and $B_{ij}$ are sent to each process $(i, j)$, the matrix $C_{ij}$ is reset to zero;
2) for each row $i$ ($i = 0, \ldots, q - 1$) of the Cartesian grid of processes, a cyclic shift of blocks of matrix $A$ is performed by $i$ positions to the left (i.e. in the direction of decreasing column numbers);
3) for each column $j$ ($j = 0, \ldots, q - 1$) of the Cartesian grid of processes, a cyclic shift of blocks of matrix B is performed by $j$ positions upward (i.e. in the direction of decreasing row numbers).

The result of such a redistribution of blocks in case $q = 2$ is shown in the figure.




Then a loop of $q$ iterations is launched, during which three actions are performed:

1) $A$ and $B$ contained in the process $(i, j)$ are multiplied, and the result is added to the matrix $C_{ij}$;
2) for each row $i$ ($i = 0, \ldots, q - 1$), cyclic transfer of blocks of matrix $A$, contained in each process $(i, j)$ of this row, is performed in the direction of decreasing column numbers;
3) for each column $j$ ($j = 0, \ldots, q - 1$), cyclic transfer of blocks of matrix $B$, contained in each process $(i, j)$ of this column, is performed in the direction of decreasing row numbers.

After completion of the loop, each process will contain a matrix $C_{ij}$ equal to the corresponding block of the product $AB$. All that remains is to send these blocks to the master process.