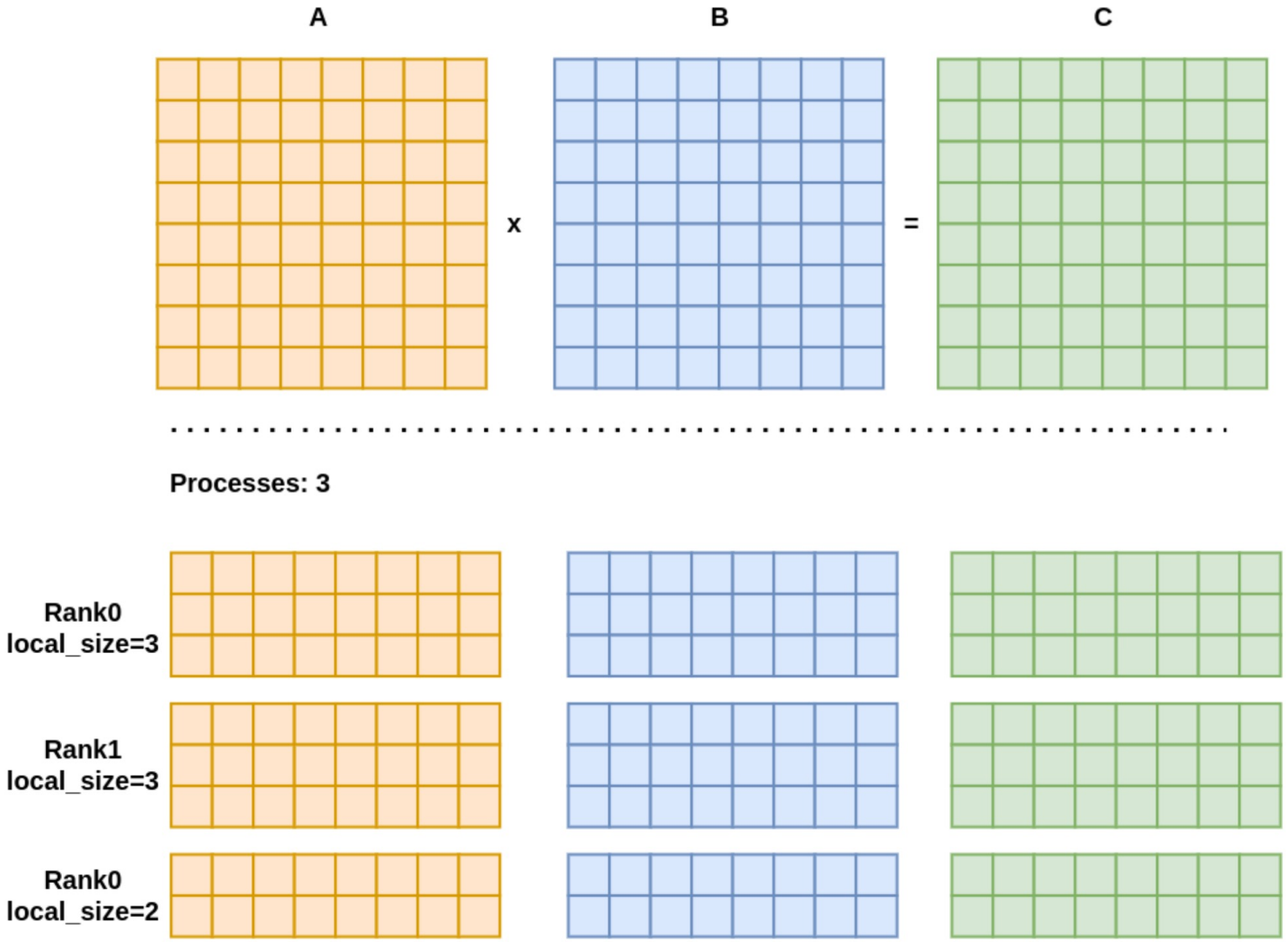
Parallel & Distributed DGEMM using Allgather

Ivan Girotto

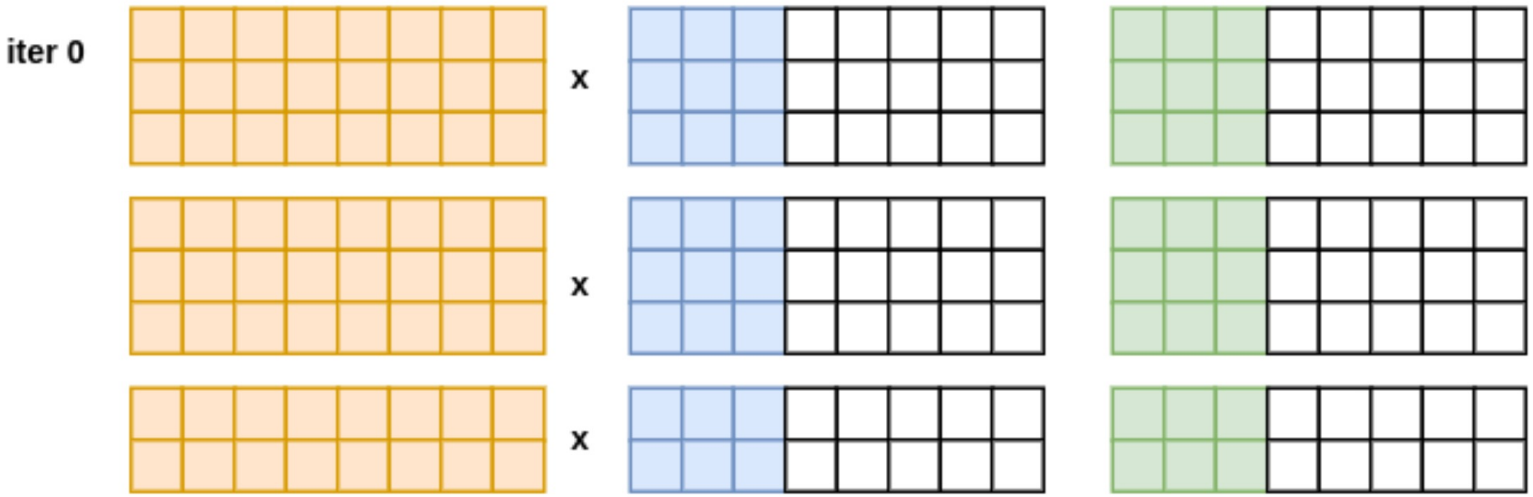
igirotto@ictp.it

All the matrices are distributed among the processing elements as represented in the following figure:



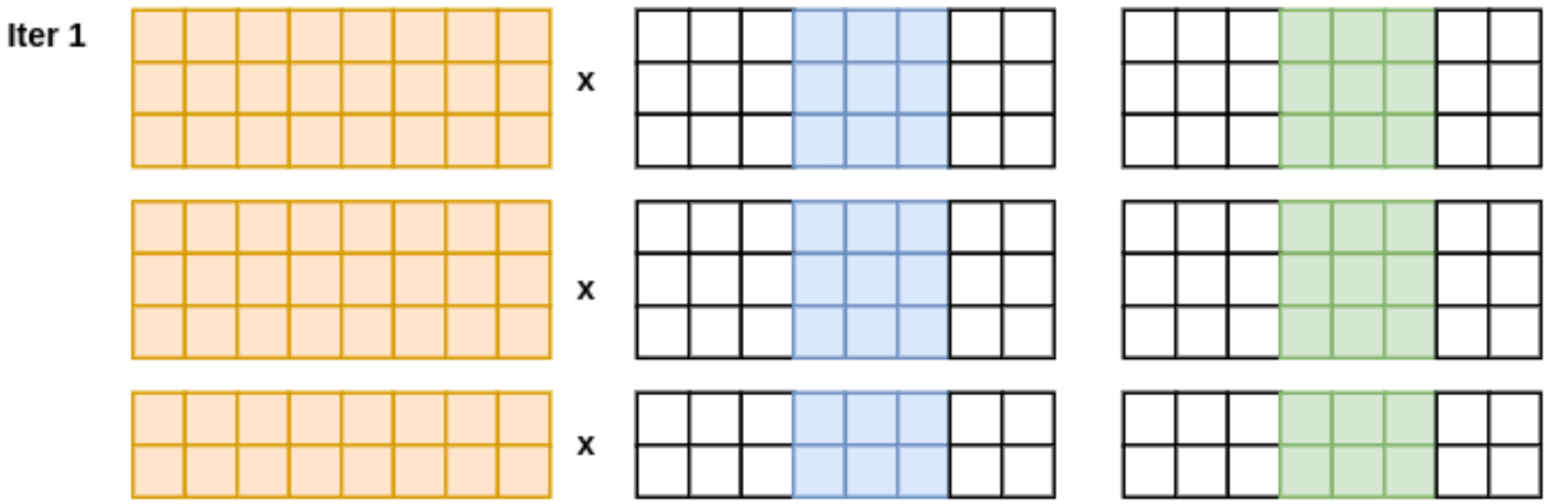
Example of 3 matrices of size 8x8 distributed among 3 processing elements The main idea is performing a for loop cycle over the number of processing elements, and at each iteration, gathering a block of columns of the matrix

Itaratively loop over the number of processing elements:



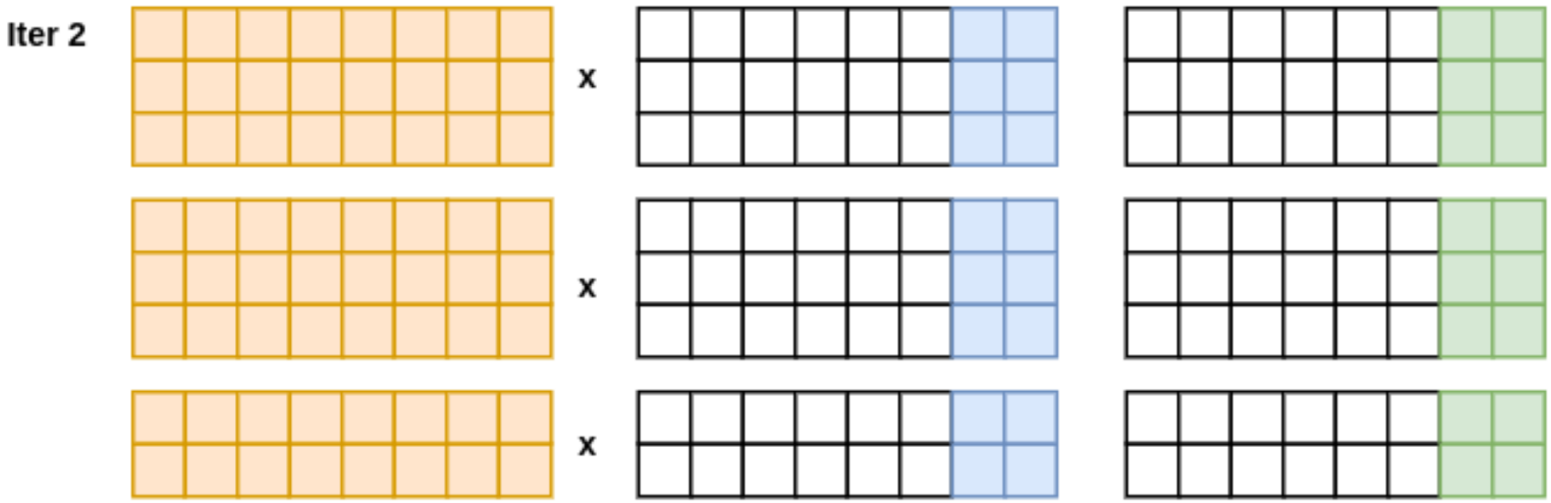
Representation of the main loop of the algorithm. Since both the size of the matrix and the number of processing can be anything, the number of elements in each block that is gathered can be different.

Itaratively loop over the number of processing elements:



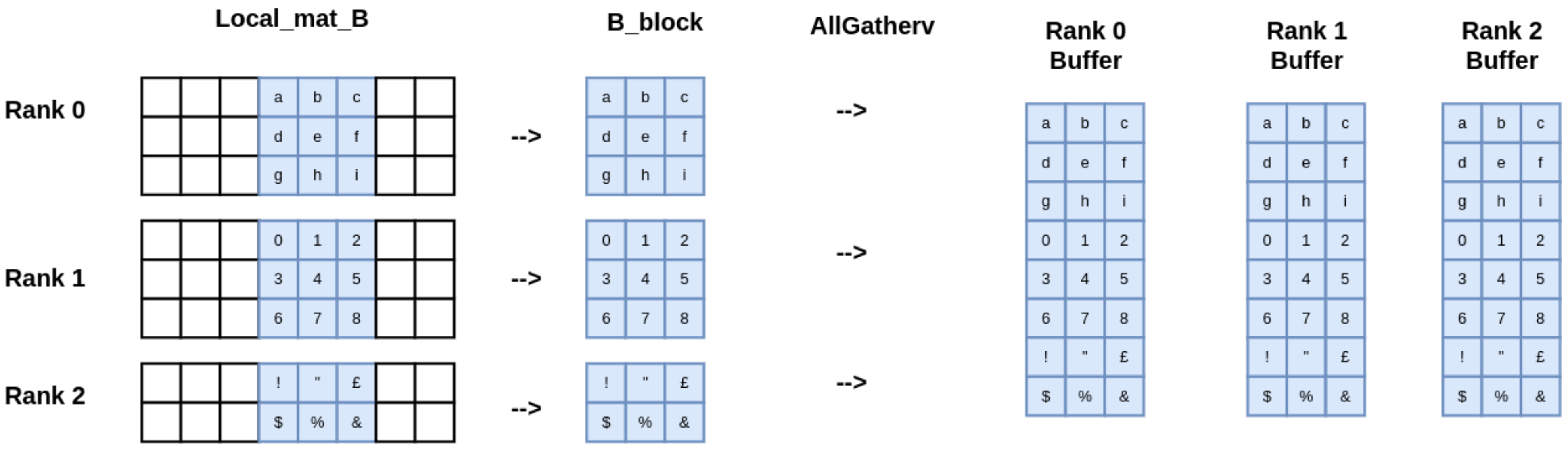
Representation of the main loop of the algorithm. Since both the size of the matrix and the number of processing can be anything, the number of elements in each block that is gathered can be different.

Itaratively loop over the number of processing elements:

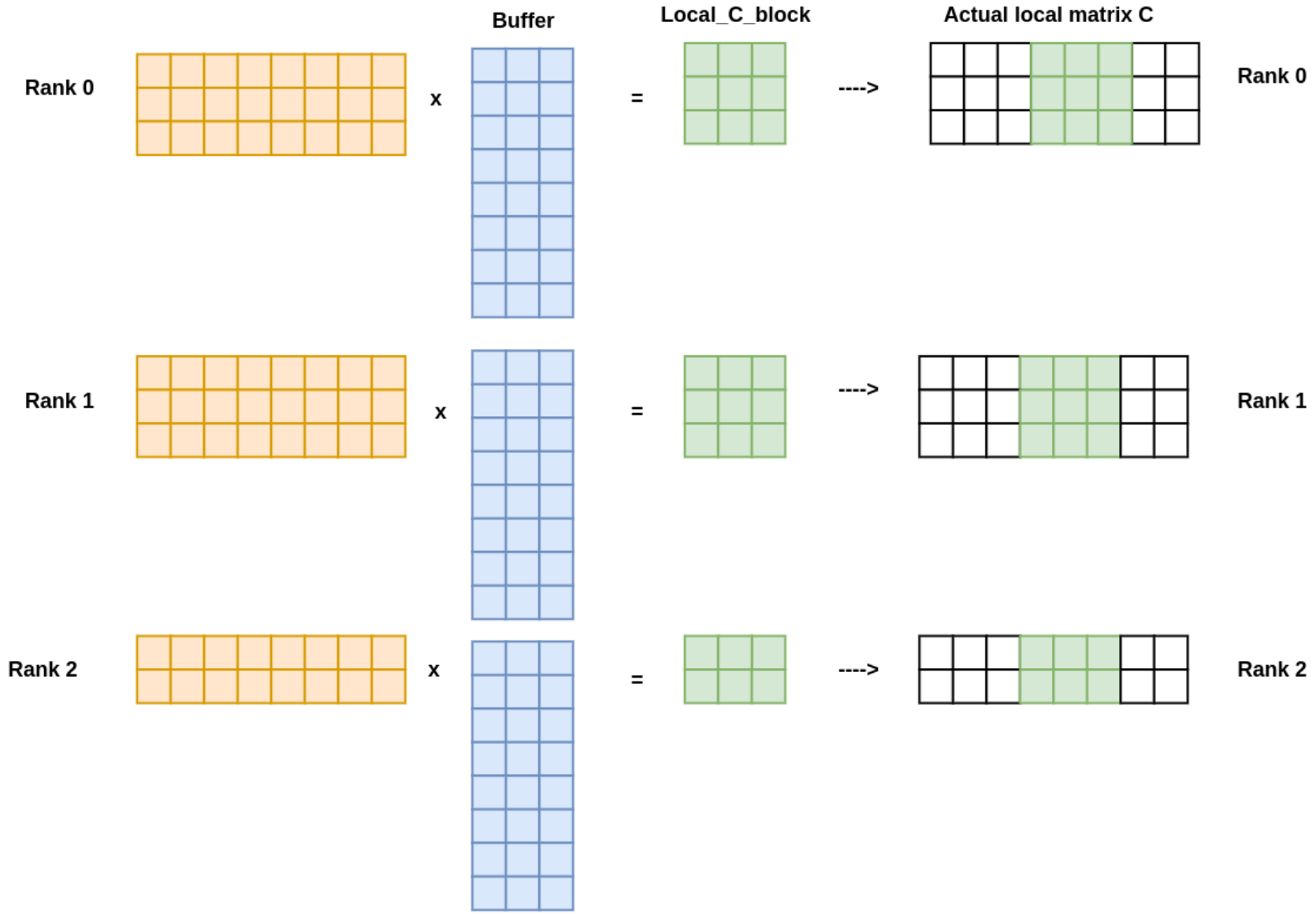


Representation of the main loop of the algorithm. Since both the size of the matrix and the number of processing can be anything, the number of elements in each block that is gathered can be different.

Block of memory to be communicated between processes are not contiguous...



Representation of how each process extract its own portion of the matrix B, and how it is passed to all the other processes with an AllgatherV operation and stored in a buffer.

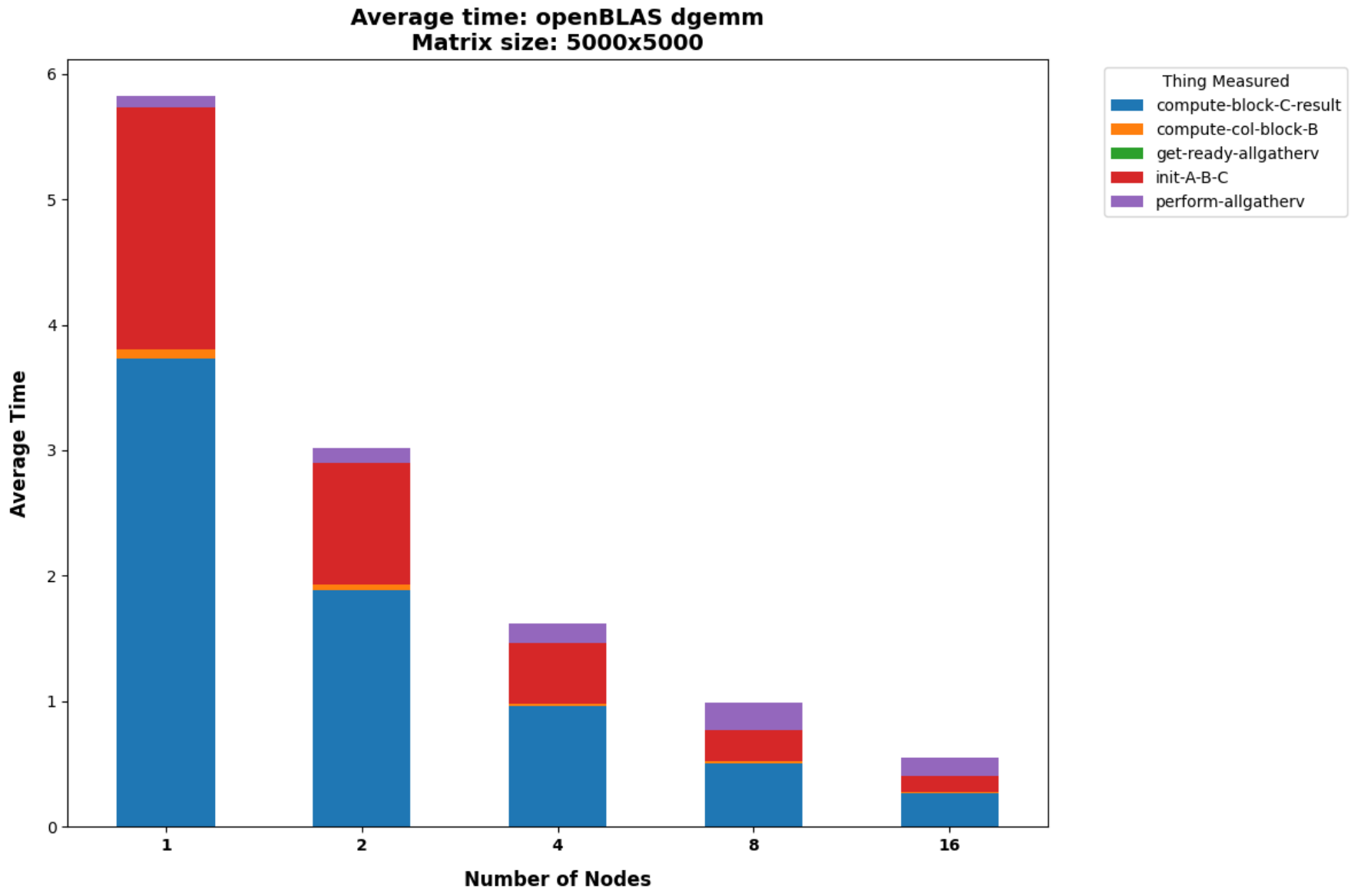


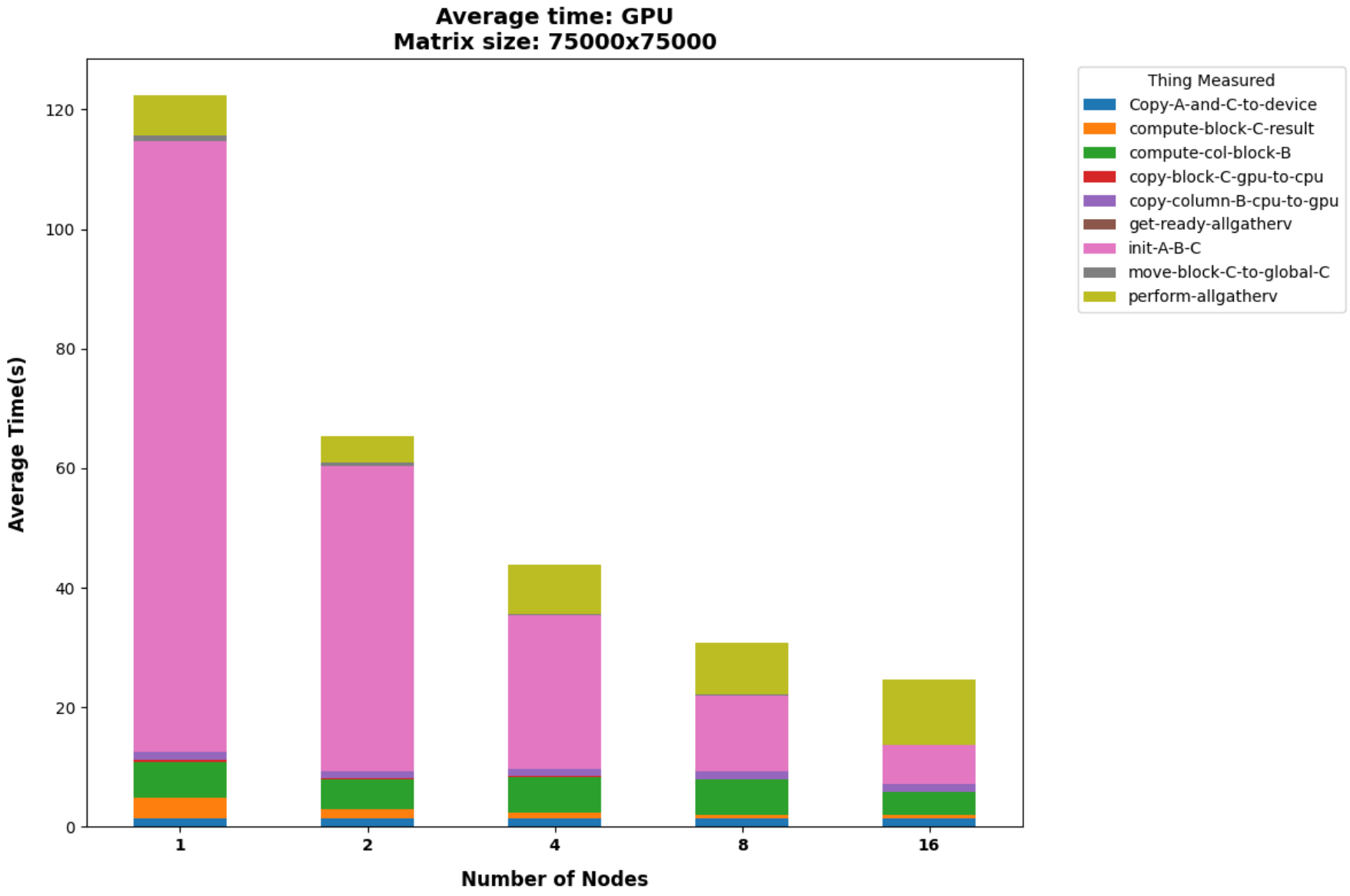
During the computation of the local portion of the matrix C, each process first compute the block of the global result. Then it is copied to the proper memory location in the global matrix C.

The product of the matrices is computed in 3 different ways:

Naive: the product is computed using the standard triple nested loop.
BLAS: the product is computed using the dgemm function of the BLAS library.
CUDA: the product is computed using the cublasDgemm function of the CUBLAS library.

Observation: There are some assumptions to make this algorithm work:
Each process has enough memory to store the local portion of the matrices.
This mean that at least every process should be able to store at list


Average time: openBLAS dgemm
Matrix size: 5000x5000
Average Time
Number of Nodes
0
1
2
3
4
5
6
1
2
4
8
16
Thing Measured
compute-block-C-result
compute-col-block-B
get-ready-allgatherv
init-A-B-C
perform-allgatherv


Average time: GPU
Matrix size: 75000x75000
Average Time(s)
0
20
40
60
80
100
120
1
2
4
8
16
Number of Nodes
Thing Measured
Copy-A-and-C-to-device
compute-block-C-result
compute-col-block-B
copy-block-C-gpu-to-cpu
copy-column-B-cpu-to-gpu
get-ready-allgatherv
init-A-B-C
move-block-C-to-global-C
perform-allgatherv