Communication-Avoiding Parallel Sparse-Dense Multiplication - Report

Piotr Karpiński

June 8, 2021

Implementation and optimizations

Program is divided into 3 parts, namely

1. `Matrix` and its subclasses: `Dense` and `Sparse`
2. `Communicator`
3. `Algorithm` and its subclasses: `ColA` and `InnerABC`

`Matrix`

`Sparse` is stored in csr format. `Dense` is stored in flat `vector` so communicating it with `MPI` procedures is fast.

`Communicator`

This part of the program handles all of the communication between processes which are

1. initial distribution and replication of sparse matrix;
2. shifts of sparse matrix between iteration;
3. `all_gathers` of dense matrices in case of InnerABC algorithm;
4. ...and many other small things.

Shifts of sparse matrices are implemented with `MPI_Isend` and `MPI_Irecv`. Sizes of sparse matrices stored in each process are `all_gathered` at the beginning of the algorithm. By doing so, each process knows exactly how big should the receiving buffer be and can start communication without waiting for information about the size of data it is going to receive. All of this allows allows for practically fully overlapped communication with computation.

Each replication group uses separate `MPI` communicator for communication within group. Each replication plane (group of processes communicating during shifts) also uses separate communicator.

`Algorithm`

`InnerABC` and `ColA` implement algorithms presented in a paper. In case of `InnerABC` synchronization, of dense matrix is implemented with `MPI_Allgather`.

Numerical intensity

For each entry in sparse matrix we need to load 2 integers (4 bytes) and 1 double (8 bytes). To multiply dense matrix by this entry, we need to read whole row. We also need to store each of the result back into dense matrix. Summing up, we execute $2n$ floating point operations, load $8 * (n + 1) + 4 * 2$ bytes and store $8n$ bytes. This gives us

$I = \frac{2n}{8n+8*(n+1)+4*2} = \frac{2n}{16n+16} \approx \frac{1}{8}$

Weak and strong scaling

All plots are based on runs where sparse matrix had 3% non-zero elements. In case of runs used to measure strong scaling, `-e 300` flag was used. For weak scaling I set `-e 1000` flag. In both cases, flag `-g` was set and `-v` was not.

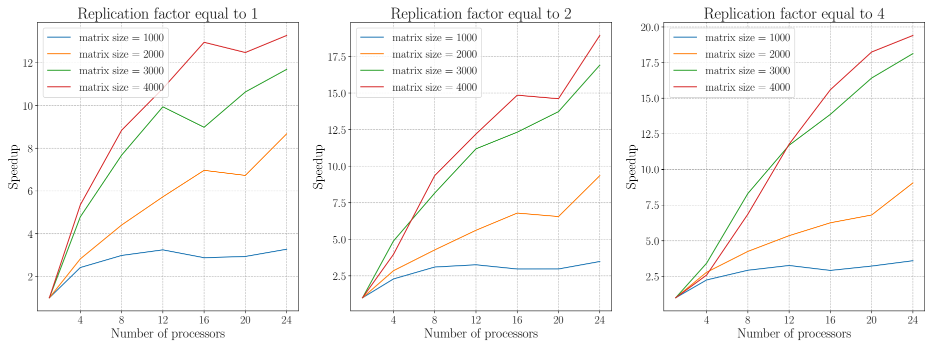

Figure 1: Strong scaling for ColA algorithm.

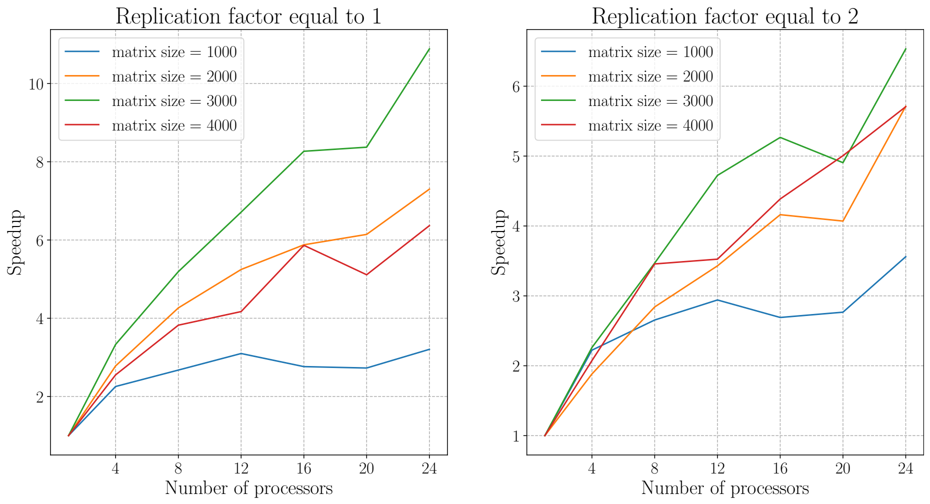

Figure 2: Strong scaling for InnerABC algorithm.

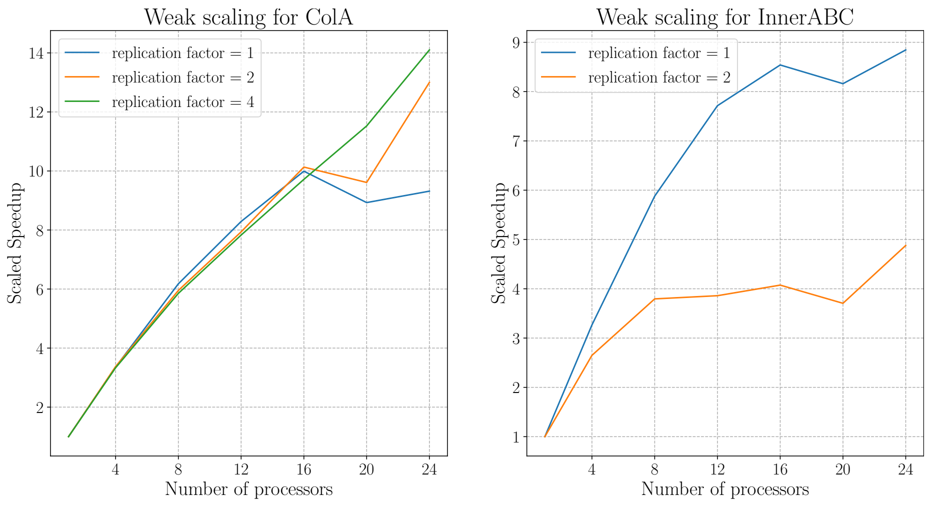

Figure 3: Weak scaling. Tested for 1, 4, 8, 12, 16, 20, 24 processors. For n processors matrix has $1000 * \sqrt[3]{n}$ rows and columns. For configurations, where replication factor is greater than 1 it is forbidden to use single processor. I used measurements from runs, where replication factor was equal to 1, and single processor was used to simulate single processors measurements for bigger replication factors.