Assignment 1

Group 7

Gaia Forghieri, Fjona Minga
Antonio Pelusi, Alberto Stefani

High Performance Computing a.y. 2022/2023

Problem: LU decomposition

Being A a square matrix (NxN), its LU factorization refers to a factorization into a lower triangular matrix L and an upper triangular matrix U:

$$A = LU = \begin{pmatrix} 1 & 0 & 0 \\ l_1 & 1 & 0 \\ l_2 & l_3 & 1 \end{pmatrix} \begin{pmatrix} u_1 & u_2 & u_3 \\ 0 & u_4 & u_5 \\ 0 & 0 & u_6 \end{pmatrix}$$

Thus, solving a linear system of equations Ax = b is equivalent to studying the following system:

$$Ly = b; \qquad Ux = y.$$

With the Gauss algorithm, A can be transformed through matrix T to obtain an upper triangular matrix:

$$TA = T_N \dots T_2 T_1 A = U$$

All T's can be chosen such that their inverse are lower triangular matrices, thus $T^{-1} = L$

Algorithm

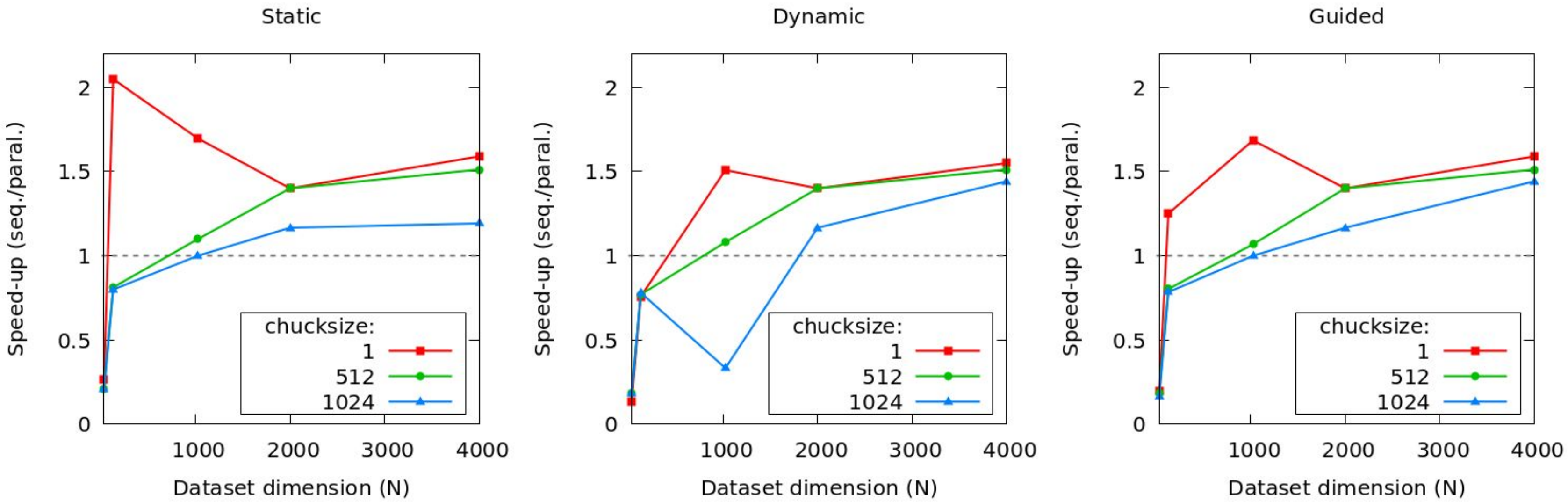
```
for (k = 0; k < _PB_N; k++)
{
        for (j = k + 1; j < _PB_N; j++)
            A[k][j] = A[k][j] / A[k][k];
        for (i = k + 1; i < _PB_N; i++)
            for (j = k + 1; j < _PB_N; j++)
                A[i][j] = A[i][j] - A[i][k] * A[k][j];
}
```

The output is a matrix containing U on the upper triangle (with the diagonal) and L-1 on the lower triangle (without the diagonal).

Inside the external loop, the algorithm performs a renormalization of each line and then performs sum on all the elements of the submatrices.

→ To avoid racing conditions we can parallelize the internal for loops.

Loop scheduling

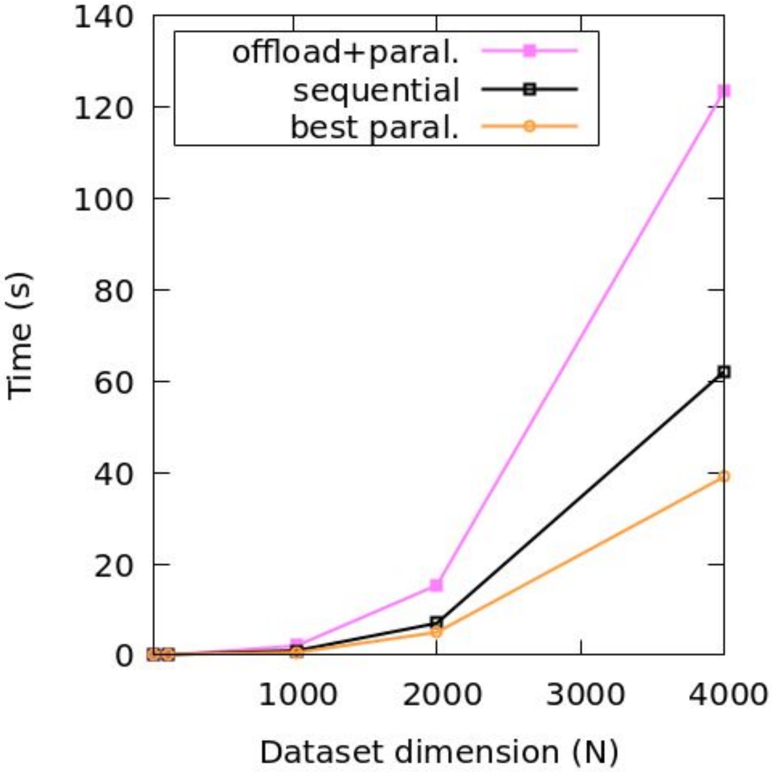


Similar behaviour for all three, slightly better for static since it requires less overhead and the loops are equally heavy.

The gain is always higher for the littlest chunksize (=1), as it guarantees a completely equal distribution of the work.

Offloading

We tried to offload work on the gpu but since the computational work is simple and the size of the dataset is huge, even if we optimize the execution a bit, the time spent to copy data from and to the gpu still takes longer than the sequential execution.



Conclusions

- Overall the best solution for this kind of problem is to use the pragma for with the static scheduling clause with small chunksize, since it balances the work with small overhead.
- We tried to use sections but it is not useful in this case: in fact, sections are needed when there are different sections of the code that work independently from each other. Same thing for tasks.
- The offloading is not a good option for this kind of problem.