Homework 2

HBM514E - Parallel Numerical Algorithms and Tools

**Note**

*The times presented below are from 10,000 square matrix instead of 50,000 since these last ones are still computing.*

*The times obtained for an error value of 10-7 are all due to hitting the maximum number of iterations, which was set as twice the matrix size (i.e., 20,000).*

*The 256 cores results could not be obtained due to the job submission being rejected.*

We can observe a quasi-linear speed-up for our LU factorization, as well as both iterative methods with high precision. With a lower precision, however, we can notice a less significative speed-up, which quickly fall once reaching 16 cores.

As for the efficiency, it looks like all our current methods of barely efficient. The factorization has an efficiency of 80% with 2 cores, and no more than 50% with more. The iteratives methods seems to be slightly more efficient when using less than 16 cores, but once again fall completely with more.

The most likely reason to this limit of 16 would be the fact that the hardware used have a maximum of 28 cores per node. This means that for 32 and more, we will not only have intra-node communication but inter-node as well. This would require reaching a deeper layer of the communication protocol, which would obviously take more time.

Another possible reason would be because both my Jacobi and Gauss-Seidel algorithm makes each process communicates with every other process. On the other hand, the factorization algorithm uses a grid layout, with every process communicating only with the others on their row and column. With waiting for less processes, it would be reasonable for the communication to be faster. Additionally, considering the initial low time of the iteratives methods, it is not surprising less improvements are possible.

The LU factorization method can be decomposed into three parts: the factorization, the forward elimination, and the backward elimination. The factorization is composed of three loops, two loops for the x and y displacement, and the last loop for the diagonal movement. The x and y loops begin at the diagonal up to the end. The two eliminations are composed of the traditional x and y loops. For that reason, the overall time complexity would be around O(N3). The communication consists of two broadcasts per iteration for the factorization, as well as a single send/receive for each elimination.

The Jacobi method can also be decomposed into three parts: the matrix-vector multiplication, the equation resolution, and the error calculation. The matrix vector multiplication is composed of a double loop, while the two other parts can be assimilated to simple loops. The communication is limited to an all to all communication after the multiplication and the error calculation. All these steps are repeated until the error is acceptable, up to 2N times, so the complexity would be O(kN2), with O(N3) when no convergence are reachable.

The Gauss-Seidel method can, again, be divided into three parts: the calculation for even cells, for odd cells, and for errors. Error calculation is still a single loop, while the two others are double loops. The communication this time occurred after the even and odd cells are calculated, using an all to all communication. For that reason, like before, the complexity would be O(kN2) or O(N3).

As for the actual results, we actually used only a dozen of iterations for the iterative methods with low precision, which is far from the 10,000 iterations used by the factorization. On the contrary, the high precision iterative methods reach the limited number of iterations, which is 20,000. The reason why the factorization is more than ten times faster than high precision iterative methods is likely due to the fact the double loops at each iteration become smaller and smaller as the number of iteration increases.