HOMEWORK 1

Dense Matrix-Vector Multiplication

Performing a matrix-vector multiplication on a parallel system using a two dimensions cartesian topology required first and foremost a square number of processors. It is therefore possible to arrange them as a grid of equal width and height. We then need to share the matrix and vector to every processes. Instead of sharing the whole objects, its is more efficient to share parts of it.

We know that all the elements on the same column of a matrix need to be multiplied by the same element of the vector. Thus, if we are to split the matrix into section of equal size to send to each process, then the same part of the vector will be used by every process on the same column.

So we first split our initial matrix and vector into a wide column matrix and subvector respectively. Then we need to send them to each one process on each column. The receiving processes will then further split the column matrix into submatrix distributed to the other processes on their column, alongside the subvector.

Once every vector has performed its own multiplication, we need to gather all the results. We know that all values on the same row need to be sum together to get the final result. So we need to send the vector resulted from the multiplication to a single process on each row of our topology, and have this process addition them together. Once this is done, collecting and combine every subvector give us the final vector of the multiplication.

While I do not know how should I plot both Gustafson and Amdahl’s laws, I do know which part of the multiplication process can be parallelized, and which can’t. Namely, it consists of the multiplication part, composed of two loops, as well as the addition part. However, this last one is also performed on two cases: one inside the two loops, and another while regrouping the processes on the same row.

Effectively, I believe the multiplication/addition part can be sped up by n times, where n is the number of processes, while the last part is sped up by square root of n. However, the parallelization will also incur an overhead, which may decrease the acceleration.