This project aims to implement the wave equation using an explicit finite difference discretisation, and demonstrate the process of wave propagation in the case of one or multiple disturbances under the different boundary conditions. Implementing the algorithm seriously is easy, but parallelling the code to improve the performance with MPI is our goal and quite a challenge.

Our algorithm contains four steps.

1. first we do communication between the processors. This step is no-blocking using Isend and Irev functions.
2. At the same time, we update the data points except for the edge data, cause those “middle” data does not require the value from other processors.
3. After that, we use waitall to wait for all communication to finish, And then we copy those data into the grid.
4. the Last step is to update the edge data and apply the boundary condition.

In theory, the optimal situation is when step 2 is finished, the communication is completed right now, so there is no time wasted waiting for the communication. Let us assume each processor task a chunk whose size is m x n, the edge data is 2m+2n - 4. When m and n are large, the edge data is A small percentage of the grid. Hence the cost of communication is less than step 2. As the growth of the number of processors, the m and n decline, fewer data need to be calculated in step2. At the same time, the degree of parallelization increases, and the overall running time of the program decreases.

However, as the number of processes increases, each process is responsible for a smaller and smaller chunk of data, resulting in each process processing less and less non-boundary data, and finally, for a process, all data is boundary data. In this case, Waitall will waste a lot of time and no calculation will be done in the second step.

In this part, we show the application efficiency, comparing the run time with setting the different number of processors and grid size. Actually, we achieve this in two ways, running the python scripts using “mpiexec -n” and “qsub *.pbs” .In that, we draw two pictures, but both of them show the same thing.

As the figures show, in general, the time cost declined when the number of processors increases. This is consistent with what we've declared before.