Imperial College London

ACSE-6.2 COURSEWORK

IMPERIAL COLLEGE LONDON

APPLIED COMPUTATIONAL SCIENCE AND ENGINEERING

Parallelisation of the wave equation in 2D using MPI

Author:
Joshua Piers Lister (CID: 01977866)

Date: April 9, 2021

1 Introduction

Solving computational problems often require high processing power. Running a program on a single system often progresses slowly, and thus there is a need for parallel programming. Open MPI is a message passing library that allows just that. This project aims to use Open MPI and various optimisations to significantly speed up the run time of a provided serial finite difference method (FDM) explicit solver for the hyperbolic wave equation in two spatial dimensions with different boundary conditions. All performance tests were run on the DUG HPC using 1 node and a varying number of processors. The tests did not include printing the grid to a file.

2 Performance

The speedup ratio is given by the equation:

$$S = T_1/T_N \tag{1}$$

where $T_1$1 is the time taken on one core (serial) and T_N for N cores. The speedup for the simulation ran under any of the boundary conditions is overlapping until 8 cores as seen in the figure 1. The Dirichlet starts to deviate, with a worsening speed-up ratio until 32 cores. This could be due to zeros at the boundary when calculation $u^n(x, y)$. Further investigation with visual studio (VS) profiler could be used to investigate the claim. At 64 cores, Dirichlet is faster than Neumann, likely due to performing fewer calculations when applying the boundary condition on each timestep iteration. The Neumann boundary condition is applied to the full perimeter of the virtual grid(imax, jmax), while the Dirichlet boundary condition is only applied where values on the perimeter are no longer zero due to receiving data from neighbouring processors.

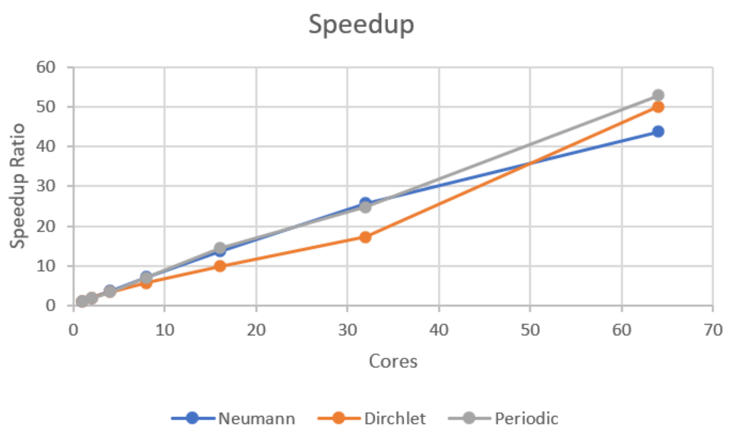

Figure 1: Speedup of the simulation with different boundary conditions on DUG HPC using a matrix size of 1000x1000 and a tmax value of 20.

As expected, parallel efficiency for all boundary conditions decreases as the number of cores increase. However, this decrease is rather small across the range of cores. All communications used are peer-to-peer,if different architectures such as master/slave were to be incorporated it would likely decrease notably faster than as is seen in figure 2. The parallel efficiency value for 64 cores with boundary condition Dirchlet and periodic doesn't follow the downward trend. Given the small sample size N=3, the timings should be repeated to rule out anomalies. Using Amdahl's law we come to the equation:

$$E \approx 1/1 + kP^{n-1}N^{1-n} \tag{2}$$

where k is problem specific and is related to the relative cost of communication and computation and $0 < n < 1$. This implies that for a given problem P, the efficiency drops as the number of cores N decreases as seen for Dirchlet, Periodic (excluding last data point) and Neumann boundary conditions.

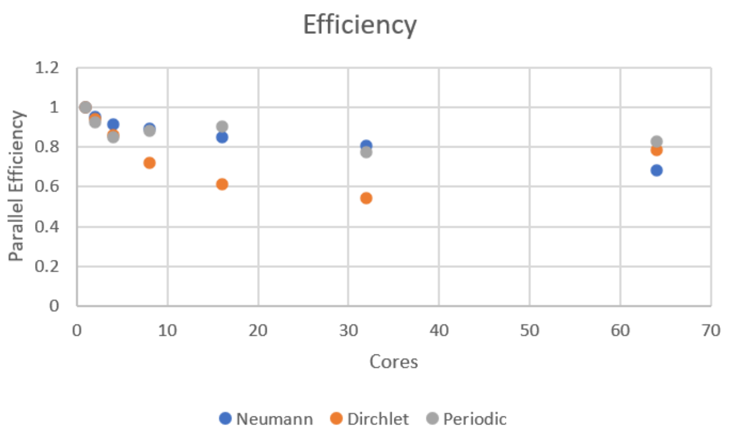

Figure 2: Parallel efficiency of different boundary conditions over a range of cores using 1000x1000 matrix size.

Across all boundary conditions, the time taken is roughly $1/N$ with an N-fold increase in processing power where N is the number of processors. While the serial time is different for the boundary conditions, the time taken at 64 processors is not dissimilar. Past 64 cores, we may expect to see Periodic becoming increasingly slower as the communication time becomes a major limiting factor for small matrices. All communications are nonblocking and work is done which is independent of the ghost cells/halo while the halo exchanges occur. The calculation time is proportional to P/N whereas the communication time is proportional to $(P/N)^n$. Thus, sending and receiving data will contribute more to parallel inefficiencies.

Table 1: Average timings (N=3) of the simulation of the 2D wave equation on DUG HPC using different boundary conditions with a matrix size of 1000x1000 and a tmax value of 20.

Cores	**1**	**2**	**4**	**8**	**16**	**32**	**64**
Dirchlet Boundary [s]	175.2	93.0	50.8	30.4	17.9	10.1	3.5
Neumann Boundary [s]	155.1	81.6	42.5	21.8	11.4	6.0	3.6
Periodic Boundary [s]	173.7	94.1	51.1	24.7	12.0	7.0	3.3

3 Scalability

The scalability is close to linear for the Dirchlet and Neumann boundary when comparing the matrix sizes 500x500 and 1000x1000. However, for matrix sizes past 1000x1000, the scaling is below par which is likely due to the communication between processors taking longer as communication time $(P/N)^n$ does not scale linearly with the number of bytes exchanged. Minimising codependency and communication between processors would improve scalability.

The periodic boundary simulation has no linear scalability for any of the matrix sizes. For smaller matrix sizes Periodic is faster than Dirchlet and Neumann, however, for 6500x6500 Periodic is the slowest.

Table 2: Average timings (N=3) of the simulation of the 2D wave equation on DUG HPC using different boundary conditions with varying matrix sizes, cores = 32 and a tmax value of 20.

Matrix size	**500x500**	**1000x1000**	**2000x2000**	**4000x4000**	**6500x6500**
Dirchlet Boundary [s]	2.7	10.1	77.4	534.0	1456.0
Neumann Boundary [s]	1.75	6.0	44.0	343.0	1443.1
Periodic Boundary [s]	0.9	7.0	49.1	353.4	1475.1

4 Design

The interface and the algorithm functions are separated into their own class files. The main function runs the appropriate simulation depending on the boundary condition and whether the number of processors is a prime number. The matrix size and physical parameters are all assigned by the user in 'input.txt'.

5 Conclusion

The serial code was improved by removing unnecessarily slow function calls, using parallelisation, boundary-specific functions, and using peer to peer communication. Speedup tests demonstrated continued linear speedup across the entire range of processors used for all boundary conditions. Parallel efficiency slowly decreased as the number of cores used increases for all three boundary conditions. The scalability for periodic was below par across all matrix sizes while Dirichlet and Neumann scaled linearly up until a matrix size of 1000x1000.

6 Future work

Test performance on non square matrices and with a prime number of processors. Compare performance when printing to grid to file and when not.