Uppsala University
Information Technology
Karl Ljungkvist

Parallel and Distributed Programming
May 3, 2017

Compulsory Assignment 2

Numerical solution of the wave equation using MPI

1 Problem setting

In this assignment, we consider the second-order wave equation with constant wave speed $c = 1$, solved on a 2D unit square with homogeneous Dirichlet boundary conditions,

$$\begin{aligned}
\frac{\partial^2 u}{\partial t^2} &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} && \text{for } 0 \leq x \leq 1,\ 0 \leq y \leq 1 \\
u(0, y, t) &= 0 && \text{for } 0 \leq y \leq 1 \\
u(1, y, t) &= 0 && \text{for } 0 \leq y \leq 1 \\
u(x, 0, t) &= 0 && \text{for } 0 \leq x \leq 1 \\
u(x, 1, t) &= 0 && \text{for } 0 \leq x \leq 1 \\
u(x, y, 0) &= \begin{cases} \cos^2\left(\frac{\pi|\mathbf{x}-\mathbf{x}_c|}{2w}\right) & \text{where} |\mathbf{x} - \mathbf{x}_c| < w \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

where the "trigonometric bell" in the initial condition has center $\mathbf{x}_c$ and width w, respectively.

By introducing a discrete grid function $u^n_{i,j}$ defined at grid points $x_i = \frac{i}{N-1}$ and $y_j = \frac{j}{N-1}$ for $0 \leq i, j \leq N-1$, we can solve this hyperbolic partial differential equation numerically using centered finite differences in both space and time, yielding

$$\frac{u^{n+1}_{i,j} - 2u^n_{i,j} + u^{n-1}_{i,j}}{\mathrm{d}t^2} = \frac{u^n_{i-1,j} - 2u^n_{i,j} + u^n_{i+1,j}}{\mathrm{d}x^2} + \frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\mathrm{d}y^2}$$

which is known as the Leap-Frog scheme. This is an explicit method, so we can solve for $u^{n+1}_{i,j}$,

$$u^{n+1}_{i,j} = 2u^n_{i,j} - u^{n-1}_{i,j} + \lambda^2 \left(u^n_{i-1,j} + u^n_{i+1,j} + u^n_{i,j+1} + u^n_{i,j-1} - 4u^n_{i,j}\right)$$

where we have used that $\mathrm{d}x = \frac{1}{N-1} = \mathrm{d}y$ and introduced $\lambda = \frac{\mathrm{d}t}{\mathrm{d}x}$. This stencil is used to compute the solution at the next time step $n+1$ based on the solution at the current and previous time steps n and $n-1$, respectively. The stencil is valid for all $0 < i < N-1, 0 < j < N-1$, whereas for the boundary points, we have

$$\begin{aligned}
u^n_{0,j} &= 0, \\
u^n_{N+1,j} &= 0, \\
u^n_{i,0} &= 0, \\
u^n_{i,N+1} &= 0,
\end{aligned}$$

at all time steps.

2 Parallelization

Your task is to implement a parallelization of the numerical solver by dividing the computational domain into a 2D partitioning of p_1 by p_2 tiles, where process (p_1, p_2) will compute the solution in tile (p_1, p_2). Since the stencil for a point i, j uses the value from neighboring points $i \pm 1$ and $j \pm 1$, the points which are located on the edge of one tile will need to access values from the neighboring tile. This can be achieved by letting each process store a slightly larger tile which includes these halo points where they are applicable. While halo points in the y dimension can easily be communicated as compact rows of values, the halo points in the x dimension can be communicated as columns by creating a new MPI datatype using `MPI_Cart_vector`. You should set up the processes in a Cartesian communicator, and either set up dimensions manually, or let MPI decide using `MPI_Dims_create`. Other MPI functions that could be of use are `MPI_Cart_shift` and `MPI_Sendrecv`. Your implementation should be restricted to the case where the number of points in the x and y dimensions are both N, but must be general enough to handle cases where $p_1 \neq p_2$, and where p_1 and/or p_2 does NOT divide N evenly.

Evaluate your parallelization by running benchmarks for several configurations of the matrix size and the number of processes. How small matrices still yield a speedup? Consider both the strong (speedup for a fixed problem size) and weak scaling of your program (speedup when resizing the problem to keep the size per process constant).

Note: In theory, a non-blocking implementation where computations are split in an interior part, which can be performed overlapped with the communication, and an external part depending on the foreign halo points, should be superior to a blocking approach. However, due to the small amount of operations and the very fast interconnect, this has a very small impact in practice on Rackham. Optionally, feel free to experiment with such a parallelization, but it is not required for a pass.

Included with the assignment is a serial reference implementation `wave.c` which you can use as a basis for your parallelization. There is a macro definition `VERIFY` which will change the initial condition such that the problem has a known analytical solution in the form of a standing wave. This enables an error estimate and the program will output the maximum error, which can be used for verification purposes. In fact, by doing a grid refinement study, you should see the expected second order convergence. To show that your program works, you should extend this to the parallel case by using `MPI_Reduce` with the `MPI_MAX` operation to compute the global maximum. However, make sure to disable this for your benchmarks since this completely changes the performance behavior of the program.

There is also a macro `WRITE_SOLUTION`, which if defined will make the program write the solution in each time step n to a file `solution-<n>.dat`, which can then be plotted using the included Matlab script `plot_solutions.m`. Optionally, you can extend this functionality to the parallel case by in each time step gathering the solution parts in process 0, which then writes the solution to disk. Again, the solution output must be disabled for the benchmarks to make sense.

3 Writing a report on the results

The report can be written in Swedish or English but you should use a word processor or a type setting system (e.g. LaTeX, Word, LibreOffice). The report should cover the following issues:

Report requirements:

1. *Problem description*, presenting the task.

2. *Solution method*, i.e. a description of the parallel implementation.
3. *Results*, presenting plots of speedup. The plots can be done using Matlab, Python Matplotlib or any other graphical tool.
4. *Discussion*, with observations and comments on the results (can also be included in 3 above).
5. *Conclusions*, with explanations of the results and with ideas for possible optimizations or improvements.

The code can be included as an appendix in the report, but preferably, it should instead be submitted as a compressed archive along with the report.

The assignments are a part of the examination, this means that you should protect your source code from others and you cannot copy the solution of others. The last day to hand in the report is **May 10, 2017**.