Exercise: Different Parallel Approaches for CFD code

David Henty, EPCC

1 Introduction and Aims

The profiling and MPI exercises are targeted at the simple cellular automaton code for traffic modeling. The traffic model has a number of advantages:

- it is very simple to understand;
- the communications structure is very simple;
- halo swapping involves sending and receiving single integers;
- it is therefore simple to modify and extend.

However, it is not very representative of real scientific calculations. The CFD model is much more realistic:

- it is a two-dimensional problem not a one-dimensional one;
- it operates on arrays of double-precision numbers not integers;
- it performs file IO to write out visualisation data at the end.

The CFD parallelisation strategy is completely analogous to the traffic model. The domain is split up across a line of MPI processes who each own a subset of the domain. Each iteration, processes have to communicate up and down with their two immediate neighbours to exchange boundary information which is stored locally in halos. Computing the total error requires a reduction operation just as required for computing the total number of cars that moved on the road.

The only significant differences are:

- the halos are now entire horizontal lines of gridpoints and not single cells (since the problem is now a 2D grid not a 1D road);
- the boundary conditions are non-periodic: processes at the extreme ends of the domain do not wrap round to the other end, so these processes only send and receive a single message (either up or down) each iteration.

2 Stopping criterion

One feature of the code not covered in the introductory exercise sheet is the stopping criterion. As previously described, the code runs for a fixed number of iterations and reports a final error. Now it can be configured to run until this error falls below some prescribed tolerance value.

To stop the simulation when it reaches a prescribed value of the error requires editing the code and changing the value of the tolerance. For example, with

```
double precision, parameter :: tolerance = 1.0e-4  ! Fortran
double tolerance = 1.0e-4; // C
```

the program now computes the error every iteration (requiring a call to `MPI_Allreduce`) and terminates if it falls below 10^{-4}(the error always decreases with increasing iterations). There is also a variable `printfreq` which determines how often the error is printed to the screen. If the tolerance is zero or negative (as originally supplied), the error is only computed once at the end when the fixed number of iterations are complete.

3 Areas to Investigate

Almost all the ideas described in the traffic model sheet apply directly to the CFD code in principle, although may be more difficult to implement in practice due to the 2D nature of the problem.

The areas where they differ are:

3.1 Profiling studies

- as well as changing the problem size (the number of gridpoints) with the scale factor, you can use a finite Reynolds number to increase the amount of computation per gridpoint;
- although the basic code only employs point-to-point communications, a dynamic stopping criterion introduces global communications;
- if you measure performance with CrayPAT you may wish to comment out calls to the IO routine `writedatafiles()` so only the calculation portion is analysed.

3.2 Halo-swapping

The boundary conditions for the CFD problem are non-periodic (fixed walls) whereas the traffic model is periodic (a roundabout).

This means that issues with halo-swapping that cause deadlock in the traffic model may only lead to poor performance in the CFD code. As a replacement to steps 4, 5 and 6 on the traffic model sheet I have supplied three alternative halo-swapping routines:

```
haloswap_early_recv()
haloswap_early_ssend()
haloswap_early_send()
```

You can swap these in and out by renaming your chosen version to `haloswap` and renaming the existing function to something like `haloswap_orig`.

What do you observe in terms of performance? Can you understand these results from the CrayPAT traces?