Parallel Design Patterns: Exercise 2a

Lawrence Mitchell

January 12, 2012

Domain decomposition

In this exercise, you will explore a simple serial decomposition of the mandelbrot computation from exercise 1. We will slice the two-dimensional complex domain in one direction. We will compute the mandelbrot in each subdomain in turn and copy this completed subdomain into the main image. The choice of which direction to slice the domain in is left up to you, but you may wish to think about which will be more efficient in terms of memory access when copying an image slice into the final complete image.

Template code

The template code is available in WebCT as `exercise2a.tar`. Unpack this in the same place that you unpacked exercise 1. You should now have an `exercise2a` directory inside the `mandelbrot` directory. Again, you have the option of carrying out the exercise in either C or Fortran: choose the appropriate subdirectory.

The template code now implements a function `compute_mandelbrot_set`. However, all this does is to wrap a computation of mandelbrot slices which you must implement.

The C template has the following code. You will need to implement both the `compute_mandelbrot_slice` and `copy_slice_to_image` functions. Stub functions that do nothing are provided, along with comments explaining what the function should do.

```
for ( slice = 0; slice < nslice; slice++ ) {
    image_slice = compute_mandelbrot_slice(slice, nslice,
                                           xmin, xmax,
                                           ymin, ymax,
```

```
                                                grid_size_x,
                                                grid_size_y,
                                                max_iter);
    copy_slice_to_image(image_slice, image, slice, nslice,
                        grid_size_x, grid_size_y);
    free(image_slice);
}
```

The Fortran template is similar, but `compute_mandelbrot_slice` does not return the `image_slice` instead it is an `intent(out)` dummy argument:

```
do slice = 1, nslice
   call compute_mandelbrot_slice(image_slice, slice, &
        nslice, min, max, grid_size_x, grid_size_y, &
        max_iter)
   call copy_slice_to_image(image_slice, image, &
        slice, nslice)
   if ( allocated(image_slice) ) deallocate(image_slice)
end do
```

Debugging problems

The easiest thing to get wrong in this practical is the domain decomposition. Perhaps a mismatch between the slice being computed and which part of the domain it is copied in to. A simple way to check for problems such as this is to colour the domain according to its slice number (rather than whether each point is in the mandelbrot set). When you run the code, you can then see exactly where each slice is being written to in the main image. With only a few slices, slice colours will be very similar, you can reduce the number of colours used to draw the final image by passing a small number to the `-i` argument: `./mandelbrot -i 4` for maximally 4 colours.