CMT106: High Performance Computing

Lab 1: Programming with OpenMP

Where: C/2.08, Queen's Buildings (second floor, central building)
When: 5-11 November 2019

Description of problem

You will be given a sequential program, laplaceSEQ.c, which solves the two-dimensional Laplace equation on a rectangle. The aim is to determine the electric field around a conducting object held at a fixed electrical potential inside a box at a fixed electrical potential of 1, with the electrical potential on the boundary being fixed at 0.

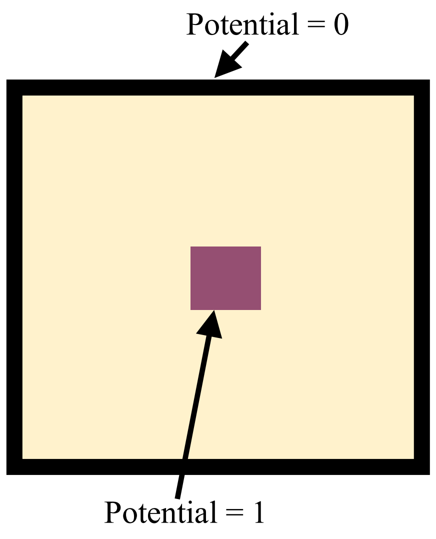

Figure 1: The conducting object is shown in purple. We want to find the electrical potential in the yellow shaded region.

The code uses an iterative algorithm to find the approximate solution at a rectangular grid of points within the yellow region in Fig. 1. The main computational work is done in the following lines of code:

```
for(k=1;k<=nsteps;k++){
    for(j=0;j<nptsy;j++)
      for(i=0;i<nptsx;i++)
        oldphi[j][i] = phi[j][i];
    for(j=0;j<nptsy;j++)
      for(i=0;i<nptsx;i++)
```

The outer loop over k is the iterative loop. This contains two doubly-nested for loops. The first copies the solution from the previous iteration (phi) to the oldphi array. The second doubly-nested loop updates the solution by replacing phi at each point by the average of the neighbouring values from the previous iteration. The mask array is set to 0 on the boundary and at the 4 central points (we don't want to change these values of phi), and is 1 at all other points.

What you must do

1. Download laplaceSEQ.c from Learning Central.
2. Parallelise the code using OpenMP. You need to include the omp.h file at the start of the code, and change the name of the output file in the output_array routine from "outSEQ.ps" to "outOMP.ps". You then need to change the lines above by adding appropriate OpenMP directives. You do not need to change any other parts of the code. The parallel code should be called laplaceOMP.c

Optional Tasks

You probably won't have time for the following tasks in the lab:

1. After the code is working correctly you should run it for different numbers of threads from 1 up to 12 and record the times taken. You will need to choose the scheduling method and chunk size: try using static scheduling and a chunk size of 20. If you have time you can try other scheduling options.
2. Plot a histogram of the time taken against the number of threads. Include a straight line to represent the time taken by the sequential code, laplaceSEQ.c. Include the problem size and number of iterations on the plot (these are the values nx, ny, and ns mentioned below). If you are not able to create a histogram then give the times in a table.

What to do if you need help

If you need help you can email me at WalkerDW@cardiff.ac.uk.

How to compile and run the code

To compile the sequential code:

```
gcc -o laplaceSEQ laplaceSEQ.c -lm
```

To compile the OpenMP code:

```
gcc -fopenmp -o laplaceOMP laplaceOMP.c -lm
```

To run the sequential code:

```
laplaceSEQ -nx 1000 -ny 1000 -ns 2000
```

This will run the sequential code on a 1000x1000 grid for 2000 iterations.

To run the OpenMP code:

```
laplaceOMP -nx 1000 -ny 1000 -ns 2000
```

The sequential and OpenMP codes should give exactly the same output files. You can check this by doing:

```
diff outSEQ.ps outOMP.ps
```

If this produces no output then outSEQ.ps and outOMP.ps are the same.

You can visualise the output using any PostScript viewing tool. For example:

```
gs outSEQ.ps
```