Distributed and Parallel Processing
Practical 1 Writeup

D.G. Smith - 13S0714

1 August 2016

1 Mandelbrot

The code changes are uppercase comments in the MandelbrotThr.java source file.

I used `ExecutorService` with a fixed thread count of five then used a `Callable` object to get the results obtained from the worker threads. A `Future` object is then used to get the actual results. I made use of this mechanism since the results of `calculateMandelbrot` needed to be obtained and passed to `display`. This ensures that the worker thread does not call `display` directly but is instead executed when the image starts being generated.

The static class `WorkerThread` implements `Callable` instead of `Runnable`. This means that `compute` is implemented which allows a result to be returned (`run` does not allow a value to be returned). This simplified the returning of the Mandelbrot result each time it is calculated.

2 QuickSort

A parallel version of quicksort was implemented by inheriting the given class from `RecursiveAction` whilst keeping it generic. A threshold variable called `WORKLOAD` is used to determine whether new threads should be spawned using `fork` and `join` or whether to just simply keep it executing sequentially.

In a test program, a `Integer` array is filled with 1 000 000 elements with values between zero and one thousand. The mean execution time for a sequential sort is 782 ms. The mean parallel sort is 259 ms. This gives a speed up of approximately 66.88%