A Work-Stealing Scheduling Technique Applied to Computing the Mandelbrot Set.

Martin J Hawes
The University of Hertfordshire

February 4, 2013

0.1 Abstract

0.2 Acknowledgements

Contents

Chapter 1

Introduction

Chapter 2

Background Research

2.1 Scheduling Techniques for Multi-Threaded Computations

This section briefly describes the problems associated with scheduling Multi-Threaded Algorithms and the major paradigms that have surfaced.

In order to efficiently utilize a parallel computer architecture it is desirable to minimize the amount of time a processor spends idle or performing other logistical tasks, I.e not doing work. When a computation's *concurrent sub-tasks* or *threads* incur a regular cost in processor time, each processor can simply have the same amount of work assigned to them. When the computation has more irregular or dynamically growing sub-tasks a problem arises that results in processors becoming idle while others still remain working. The solution to this problem is refereed to as *load balancing* and can be described as a form of dynamic scheduling that ensures each processor spends approximately the same amount of time working. This means processors generally spend less time idle, however have to deal with scheduling overheads as a trade-off.

When considering the scheduling of multi-threaded computations, two major load balancing techniques have been used. These are *work-sharing* and *work-stealing*.

- **Work-Sharing:** A thread which creates new threads attempts to migrate these to another underutilised processor at creation time.
- **Work-Stealing:** A processor that is starved of work, attempts to "steal" work from other processors.

Both techniques intend to promote balanced work-load across all processors, however in Work-Stealing the frequency of thread migrations is lower. When all processors have a high work-load and no need to "steal" this becomes useful because threads need not get migrated at all. With work-sharing thread migration occurs each time a new thread is spawned [2, p. 2]. This also suggests that work-stealing promotes better locality and grouping of sub-tasks, as spawned threads stay with the same processor until stolen.

2.2 The Work-Stealing Technique - Described in Depth

2.3 The Mandelbrot Set

The Mandelbrot set is a set of complex numbers which when plotted produce a spectacular and recognisable shape. It is often presented as a colourful and striking image and has been described by some as the most beautiful object in all of mathematics [3, p. 234]. The complex numbers that comprise the set are closely related to *Julia Sets*. In-fact the Mandelbrot set can be described as a catalogue of Julia Sets which, when plotted, all points are *connected*, forming a *single, unbroken shape* [1, p. 177].

The set is named for the mathematician *Benoit Mandelbrot*, who discovered it in 1980 [5] [1]. He was a pioneer in the study of fractal geometry and also coined the term *fractal*, of which both the Mandelbrot set, and Julia sets are examples of.

In this section I will give a more detailed explanation of the areas mentioned above.

2.3.1 Fractals and Self-Similarity

A fractal is a means of describing shapes more complex than classical Euclidian shapes. The leaves of a pine tree or the forks of a lightning bolt are obvious examples of real things that fractals allow us to more faithfully describe. These *real world fractals* are similar to *mathematical fractals*, of which the Mandelbrot set is an example, but differ in that they do not display the property of *Scale Invariance*.

The underlying property of these fractals (but not all) is some level of *Self-Similarity*, where the shape is comprised of smaller "copies" of itself. This is known as *Exact Self-Similarity* and means that the shape is identical at any scale. A simple example of this is the *Triadic Koch Snowflake* which is demonstrated in Figure **??** showing the fractal developing over three iterations of its construction. It is important to note here that The Mandelbrot set does not quite show the same property, it is said to be *Quasi Self-Similar*. This means that the shape is approximately similar at all scales, in that the shape is repeated but in a slightly distorted form with each "copy".

It turns out there are many rather useful applications for fractals. To name a few; computer game graphics,

2.3.2 Julia Sets

To understand the basis of the Mandelbrot set it is first necessary to understand it's relation to *Julia Sets*. The function in equation 2.1 is iterated over where c is fixed. The *filled Julia Set* is comprised of all *f(x)* where the result does not tend towards infinity.

$$f(x) = x^2 + c \tag{2.1}$$

2.3.3 Computing the Mandelbrot Set

2.4 Programming Tools

Chapter 3

Main Chapters

3.1 Design of the Algorithm

3.2 The Implementation

3.2.1 An Algorithm to Compute the Mandelbrot Set

3.2.2 A Concurrent Algorithm to Compute the Mandelbrot Set

3.3 Features for Demonstration

3.4 Validation and Verification

Chapter 4

Discussion and Evaluation

4.1 Analysis of the Algorithm

4.2 Reflection on Project

Chapter 5

Resources

Bibliography

[1] Micheal F Barnsley, Robert L Devany, Benoit B Mandelbrot, Heinz-Otto Peitgen, and Dietmar Saupe Richard F Voss. *The Science of Fractal Images*. Springer-Verlag, 1988.

[2] Robert Blumofe and Charles E Leiserson. Scheduling multithreaded computations by work stealing. Technical report, MIT Laboratory for Computer Science, 1994.

[3] David Feldman. *Chaos and Fractals: An Elementary Introduction*. OUP Oxford, 2012.

[4] Benoit B Mandelbrot. *Fractals: Form, Chance and Dimension*. W.H.Freeman and Co Ltd, 1977.

[5] Benoit B Mandelbrot. *The Fractal Geometry of Nature*. W.H.Freeman and Co Ltd, 1983.

[6] Jason McGuiness. Atomic code-generation techniques for micro-threaded risc architectures. Master's thesis, University of Hertfordshire, 2006.

Chapter 6

Appendices