An Investigation of Chaos Theory Using Supercomputing Techniques

Bryan Ward

March 30, 2007

Abstract

Chaos theory is the study of dynamic systems in which small differences in the environment, can create large, unpredictable results. The classic example of chaos theory is the Butterfly effect, or the theory that a Butterfly flapping can effect large scale weather patterns such as tornadoes and hurricanes even from hundreds of miles away. Chaos Theory is also applicable in systems other than weather, such as the stock market and physics. While these are very complex systems, there are chaotic mathematical systems represented by fractal images which this project aims to investigate. In such complex systems, the use of a supercomputer can be valuable in predicting results, so this project will use the supercomputer as well.

1 Introduction

1. Problem Statement
   To generate and analyze several different fractals to gain more insight into the complex topic of chaos theory. To do so using a supercomputer to learn more advanced algorithms.

2. Purpose and Goal
   The purpose of this research project is to implement supercomputing algorithms while investigating the complex topic of Chaos Theory. I chose this field for my project because Chaos Theory as fascinated me for years. This subject lends itself to the multi-processor supercomputer, and it is also a goal to learn some supercomputing algorithms.

The project itself is going to be broken up into several different things. I am starting by coding and investigating the Mandelbrot and Julia set fractals. This is relatively easy and is a good place to start to learn the Message Passing Interface (MPI). I plan to move into different fractals and chaotic attractors shortly.

2 Previous Work

Chaos Theory has many applications to numerous different real world. Lorenze coined the phrase Chaos Theory while studying meteorology because there are so many different variables that can have significant effects on weather patterns, even if they are only minor initial changes. Chaos is also evident in physics. One example could be letting a full balloon loose, the result is an unpredictable flight. This is because the countless air molecules can have differing effects, and the trajectory of the balloon is therefor unpredictable. Some more recent research has been found showing that the structure of arteries in the human body is chaotic and may be coded as such in the DNA. Brain and heart functions may be chaotic as well.

Fractals have applications outside of the physical sciences as well. In computer science, there is work being done to use fractals to generate more realistic computer graphics. This not only creates more life like graphics, but also speeds up the rendering process. Work has also been done to use fractals to compress data, and especially images, using fractals.

3 Procedure

To investigate Chaos theory, I am going to start by analyzing more simplistic mathematical fractals. Because this is also a Computer Systems Lab project I am also planning on incorporating the use of the school supercomputer to learn about, as well as take advantage of the processing power of a multiprocessor system. I am starting with a simple single processor generation of the Mandelbrot and Julia Set Fractals, and I am going to expand the use MPI and the supercomputer other rendering algorithms and coloring algorithms.

4 Progress

This quarter I have made significant progress. I have gained access to two system with MPI, the school Cray SV1, and the dual, dual core Opteron system, bottom. I have used both of these systems to test and run my programs.

At the beginning of the quarter when I did not have access to either of these systems, and thus no access to MPI at all, I wrote a new fractal rendering algorithm commonly called the buddhabrot. This algorithm is far more processor intensive, and the quality of the resultant fractal is highly dependent on the number of iterations, which is directly related to processing time. Using MPI on this program will be especially useful for just that reason.

I began to use MPI midway through the quarter when I gained access to bottom. I started by generating many Julia set frames to string together into a video. This did not require me to pass information between processors, so it was a good way to start to learn how MPI works.

Next I ran the Mandelbrot set fractal with MPI, by splitting the fractal into segments to be passed to each processor. This algorithm significantly increases performance, but there is some slow down in the time to pass the messages between the different processors.

I also changed how my program write the fractal to a file. I used to print the data to a *.pgm file in ascii, which is terribly inefficient, both in writing time, reading time, and file size. I changed this to be a binary output of an unsigned character which decreased the file size significantly.

5 Testing and Analysis

There are two ways for me to test and check my program. The first and most obvious is to check to see if the fractal image was generated properly. The other way I test the program is to time the runtime and compare to other results.

Much of my work this quarter has been working to rewrite my previous programs using MPI. Testing these programs is very simple because I can check my results directly against their single processor counterpart.

After I check that my programs are running properly in MPI, I begin to test that they work with different numbers of processors. I have tested now on both bottom and the Cray using anywhere from 2-16 processors. I have

noticed that especially on the Cray, the more processors the program is run on, the more time it takes for messages to be passed between the processors. When a 12800x10240 pixel Mandelbrot fractal was generated on the Cray using 16 processors, it took approximately 20 seconds on average for each processor to iterate through it's respective segment, but the entire program took 83 seconds to run. This means that it took over a minute to pass the data between the processors. This was still significantly faster than running it on a single Cray processor which took 365 seconds.

6 Conclusion

In conclusion, using MPI and multiple processors the runtime is decreased. The runtime is not reduced by the factor of the number of processors used, because the processors must send messages to one another which can be time consuming. Passing large amounts of data between processors takes a substantial amount of time. Algorithms which utilize more processors but reduce the amount of information needing to be passed between the processors are the most efficient using MPI.

References

[1] Bourke, P., *An Introduction to Fractals.* Retrieved February 13, 2003, from http://astronomy.swin.edu.au/ pbourke/fractals/fracintro

[2] *Chaos Theory: A Brief Introduction.* (n.d.). Retrieved January 24,2007, from http://www.imho.com/grae/chaos/chaos.html

[3] Devaney, R. L. (n.d.). *The Fractal Geometry of the Mandelbrot Set.* Retrieved February 7, 2003, from http://math.bu.edu/DYSYS/FRACGEOM/FRACGEOM.html

[4] Donahue, M. J., III (n.d.). *An Introduction to Mathematical Chaos Theory and Fractal Geometry.* Retrieved February 7, 2003, from http://www.duke.edu/ mjd/chaos/chaos.html