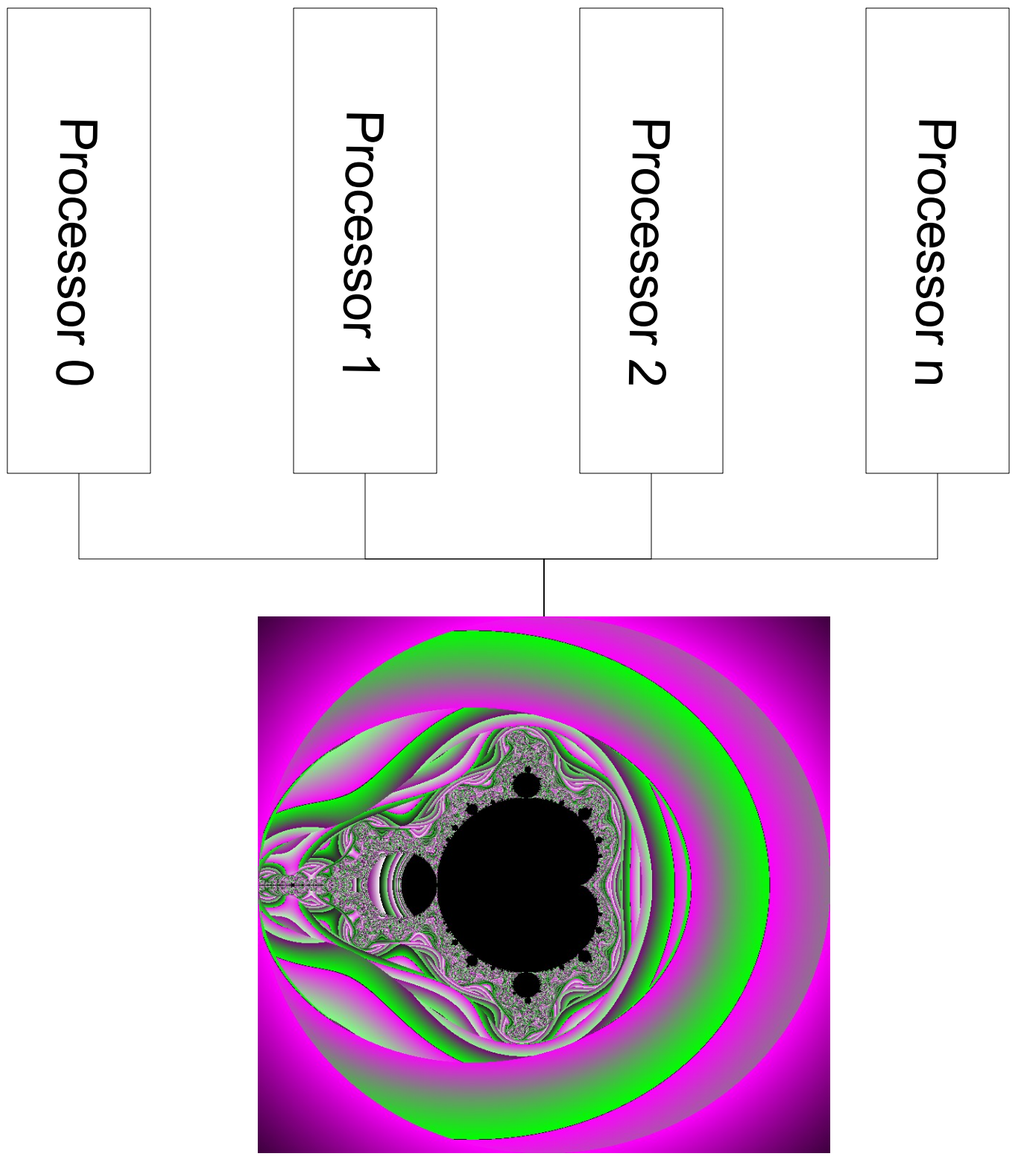
An Investigation of Chaos Theory Using Supercomputing Techniques

TJHSST Computer Systems Lab 2006 - 2007

Bryan Ward

Introduction

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 it's wings 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 this project distributed computing will be used to investigate the these fractals.



Pseudo Code:

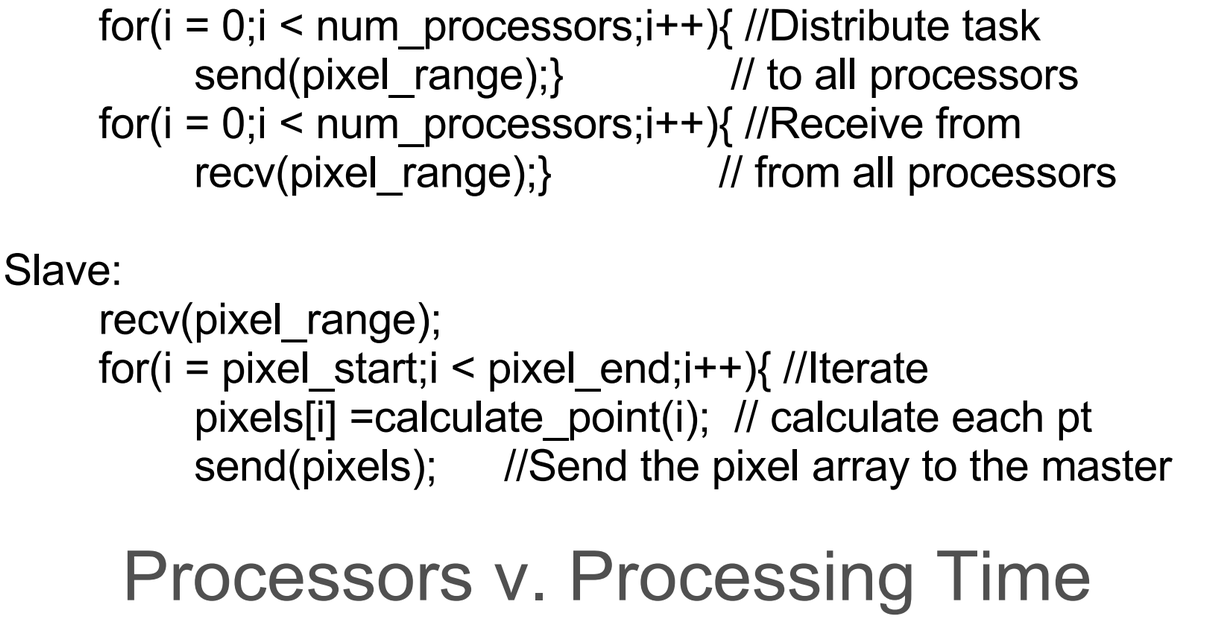
```
Master:
        for(i = 0;i < num_processors;i++){ //Distribute task
                send(pixel_range);}              // to all processors
        for(i = 0;i < num_processors;i++){ //Receive from
                recv(pixel_range);}              // from all processors

Slave:
        recv(pixel_range);
        for(i = pixel_start;i < pixel_end;i++){ //Iterate
                pixels[i] =calculate_point(i);  // calculate each pt
                send(pixels);      //Send the pixel array to the master
```

Results

MPI was used to distribute the computational load to multiple processors. The slave processors then calculated the pixel values and sent the resulting array back to the master to be written to the file (See Pseudo Code). When passing messages between multiple processors, time is spent in communication. This explains why the overall speed for twelve processors is not twelve times faster than one.

In this project I was also able to investigate different coloring and rendering algorithms and their performance in a distributed computing environment. The use of the Hue – Saturation – Value color space was used in these coloring algorithms. The output of the program was written to a binary image file for faster write speeds and smaller data files than ASCII.



Computational time:

$$t_p = \frac{iterations * n}{p} + (p-1)(t_{data}) + k$$