PA2 – Mandelbrot Documentation

Brian Conway

13 Feb. 2017

CS415

Methodology

The file sequential.cpp uses nested for loops to calculate colors of pixels in an image based on the Mandelbrot set 10 times, timing the entire time it takes to calculate the pixel colors. It divides the total time taken for all calculation by 10 to get the time taken for a single calculation of the Mandelbrot set. The dimensions of the image and the overall measurements taken can be adjusted by changing the two constants at the beginning of sequential.cpp.

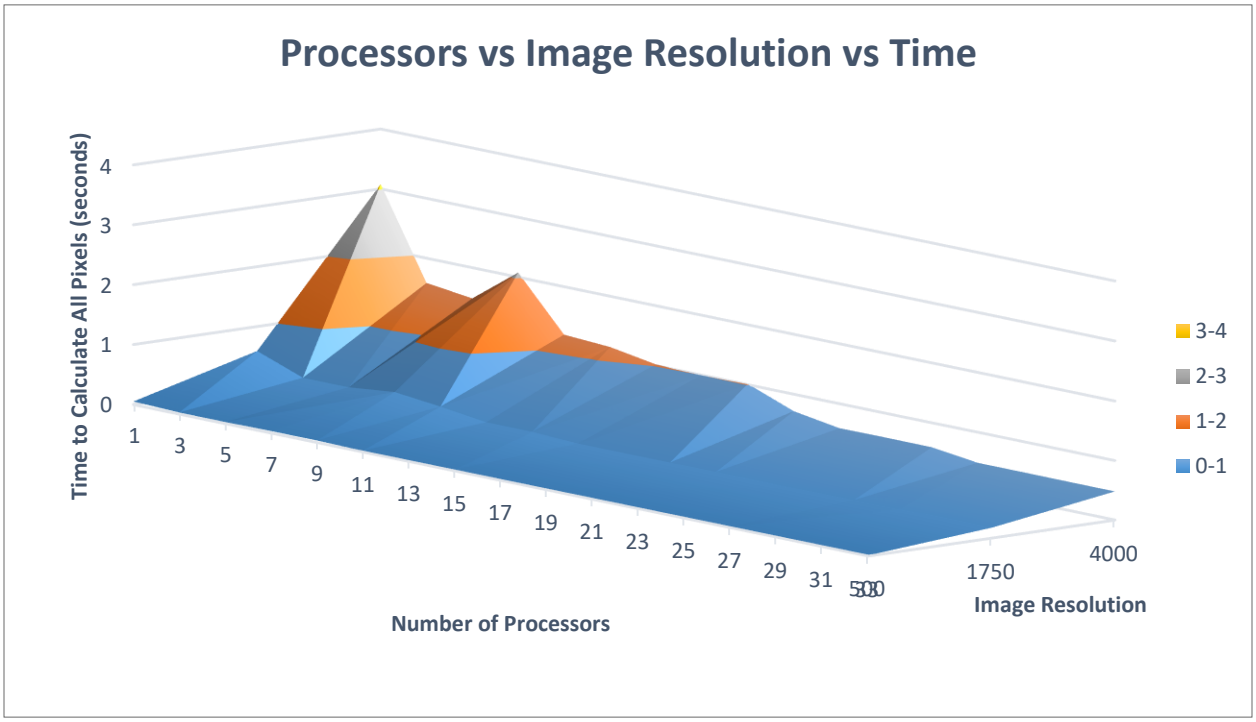
After the 10 readings are taken, the calcStatistics function finds both the average and the standard deviation of the pixel color calculation times. It outputs this information to the console and then outputs the statistics to a file called “measurements.csv”. The first line of the output file is the average, the second is the standard deviation. The image is output to the file “imageSeq.pim”

The file static.cpp uses static task assignment to calculate colors of pixels in an image based on the Mandelbrot set 10 times, timing the entire time it takes to calculate the pixel colors. It has the master node divide all the rows in the image into sets based off of the amount of tasks, then sends out the row number of the first row to calculate to each corresponding task. The other tasks receive their row numbers and calculate the pixels for that row and the subsequent rows based on how the rows were initially allocated. They send back the entire set of rows as a 1D array, which the master receives and stores it in the corresponding locations in its 2D array of pixel colors. It divides the total time taken for all calculation by 10 to get the time taken for a single calculation of the Mandelbrot set. The dimensions of the image and the overall measurements taken can be adjusted by changing the two constants at the beginning of static.cpp. The image is output to the file “imageStat.pim”

Timing is done using the custom Timer class, which has start, stop, and resume functions similar to a stopwatch. Timer makes use of timevals and the gettimeofday() function in order to get an accurate reading of seconds and milliseconds at certain instants. When the elapsed time is to be given, the Timer takes the difference in seconds and milliseconds between when the timer was started and when the timer was stopped. It then combines the seconds and milliseconds reading into a double precision floating point variable.

Results

Below is a surface graph of the amount of processors, the time taken to calculate all the pixel colors, and the three resolutions of image used. The part where it says 1 processor is the time for the sequential implementation. In general, as the size of the image increased so did the total calculation time. As the number of processors increased, the execution time generally decreased. Though there was much more speedup with the 4000x4000 resolution than the 500x500 or the 1750x1750. The graph is slightly messed up where the processors axis and the resolution axis meet. That point on the graph represents 33 processors used and a 500x500 image resolution.



Below is a mapping of the same data that makes it more clear how much speedup the 4000x4000 resolution got compared to the other resolutions. Both of the lower resolutions stop gaining much benefit after 17 processors, but the largest resolution stops gaining much speedup after around 29 processors.