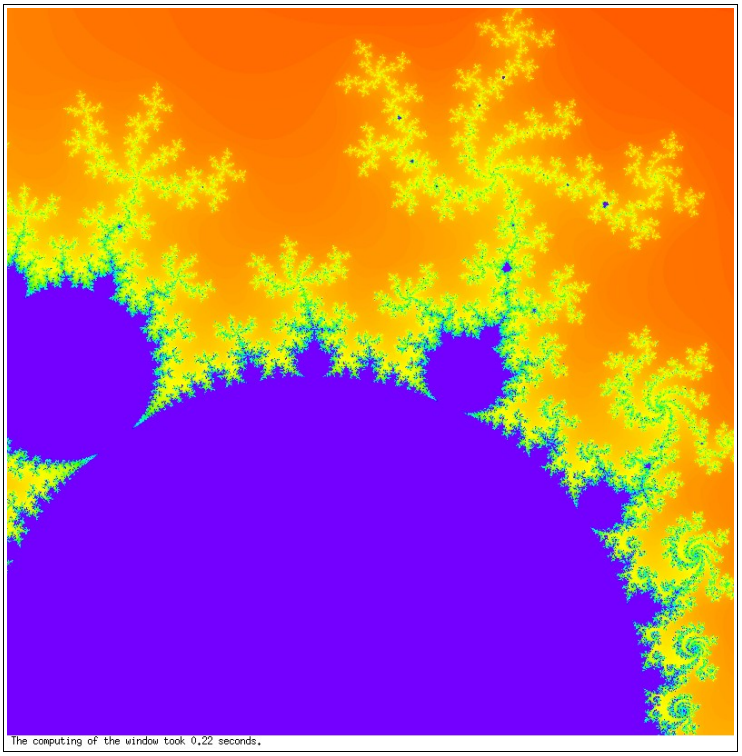
Markus Karjalainen
Mat.nr. 29849

Solution description

I have written a MPI program that calculates the Mandelbrot set and presents the result graphically to the user. The program also allows the user to zoom in on the window by selecting an area with the mouse. The program prints the time it took to compute the pixels in the footer of the window as illustrated in the image below. The program terminates if the user selects an area that has an x-axis that is less than 10 pixels. The program cannot be run on a single process, since it requires at least one master and one slave process.



I started the design process by writing a sequential version of the program and then modified it to a parallel version with a master-slave computation.

The master process, zero, is responsible for interacting with the user and displaying the graphical output whereas the slave processes, 1 – *n*, are responsible for the calculating the Mandelbrot values.

The master process initiates the calculation of a window by sending a row number, starting from row zero, to each slave, 1 – *n*, after which the master process receives rows from the slaves and sends new rows to the slaves until all rows have been computed. The master process collects the calculation results into a two-dimensional array and no graphical output is drawn until all results have been received from the slaves. I made this design decision to avoid unnecessary bias, caused by the graphical output, in the time measurement of the Mandelbrot calculations. The slave processes send the results of their computations to the master process in an array of size *window width* + 1. The number of the calculated row is placed in the last additional array index and it is utilized by the master process to determine which row the given result array refers to. I made this design decision to simply the mechanism of keeping track of the calculated rows in the master process. The returning the row number in the result array from the slave processes also enables the possibility to receive the rows in the master process in any order, since the row number information is carried with the results.

When a window has been computed totally, an end of window signal is sent to the slave processes, which causes them to stop the Mandelbrot calculations, the master process renders the graphics to the graphics window and waits for the user to select a new area to be computed. If the user selects an area that has an x-axis that is less than 10 pixels, a stop signal is sent to the slave processes, which causes them to terminate, and the master process is terminated. If the user selects an area that has an x-axis that is greater than 10 pixels, the coordinates of the area are sent to the slaves, which compute new complex *min* and *max* values and scaling factors, and a new calculation process is initiated by the master process.

The benefit of the parallel computation is clearly noticeable. When executed with 2 processes (1 master and 1 slave) on *maximum.cs.abo.fi*, the computation of the initial window of size 800 x 800 pixels took 0.16 seconds. When executed with 20 processes (1 master and 19 slaves), the computation of the initial window of size 800 x 800 pixels took 0.06 seconds. The calculation speed improved by a factor of 3/8 in the latter case. The calculation time of course depends on the window size. The window size can easily be changed by modifying the values of the *W* and *H* variables on lines 54 and 55, but $W = H$ should always hold.

For more details about the implementation, please see the attached C source code *mandelbrot.c*.