Computing a Movie of Zooming into a Fractal

Martin Burtscher, Department of Computer Science, Texas State University

I. Assignment Overview

This project is to parallelize a provided serial program that computes multiple images of a fractal, each at a higher zoom level. The resulting images can be viewed individually or combined into a movie, e.g., with the free *convert* tool.

The part of the code that needs to be parallelized is short and relatively simple. It consists of three nested *for* loops. The outer loop iterates over the images and the inner two loops iterate over the *x* and *y* coordinates. The *do* loop that computes a pixel's gray-scale value comprises only six statements but is complex, which is why I explain it in class before handing out the project.

II. Target Audience and Context

I have used this assignment for many semesters in a senior-level undergraduate course on parallel programming as well as in a masters-level graduate course on parallel processing. In both courses, the fractal is part of multiple bi-weekly programming projects. Each project focuses on a different parallelization approach (MPI, Pthreads, OpenMP, and CUDA). In addition, I typically include at least one variation per project.

III. Interesting Variations

In the most basic version, only the outer image-loop is parallelized. However, the project can be made more challenging by including some of the following additions. 1) If the zoom factor is computed iteratively (rather than using a closed-form function), there is a loop-carried dependency that the students must eliminate. 2) If the loop does not execute enough iterations to yield sufficient parallelism (e.g., for GPUs), multiple loops must be combined and parallelized together. 3) If the order of the inner two *for* loops does not match the memory layout of the image, the loops must be swapped to improve locality and enable coalescing. 4) Depending on the selected fractal, there may be substantial load imbalance, which can be alleviated by modifying the schedule.

IV. Prerequisites

At a minimum, the students must have been taught basic loop-parallelization strategies. If the above-mentioned variations are used, knowledge about dependency elimination, loop fusion, loop interchange, and scheduling techniques is also required.

V. Covered Concepts

The covered concepts include basic loop parallelization and load imbalance. Optionally, they include which loop to parallelize (to avoid dependencies or to lower overhead), loop interchange (to enhance locality), loop fusion (to increase parallelism), dependency elimination, and scheduling (to reduce imbalance).

VI. Strengths

I have had great success with this assignment because of the following benefits. 1) Most students enjoy creating the fractal image/movie. 2) The code is quite short but not trivial, and the part that needs to be parallelized easily fits on a single screen. Not counting writing out the BMP, there are only about 40 statements, including timing code and code for checking the command-line parameters. 3) The program requires no input files, just the size of the image and the number of images to compute. 4) The output can be viewed image-by-image or converted into a movie, the latter of which is great for demonstrating the result to other people. 5) The images are not only visually pleasing but also useful for debugging, e.g., they typically expose which thread or process is off. 6) The fractal can easily be changed from one semester to the next to modify the runtime and the load imbalance.

Probably the most distinctive feature of this assignment is that the brightness of each pixel is determined by the amount of computation performed. Hence, the fractal is a visualization of the workload and thus enables the students to see the load imbalance. For example, when generating Figure 1 with two threads, there will be substantial load imbalance if one thread computes the less work intensive, brighter top half of the image and the other thread the more work intensive, darker bottom half.

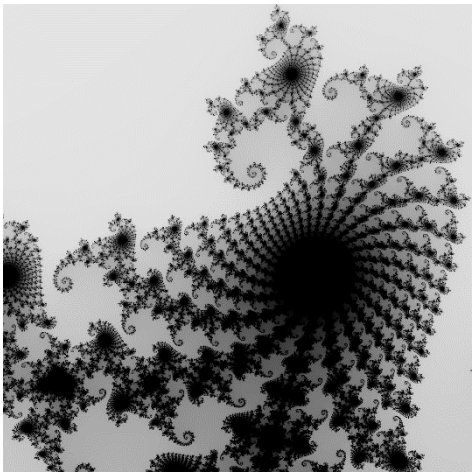

Fig. 1. A frame of the fractal movie

VII. Weaknesses

The main downsides of this project are that the computation is not particularly useful, that it is hard to understand (though asking students to parallelize code they do not grasp completely may still be useful as this situation does occur in practice), that the provided BMP-writing code is not pretty because of various quirks of the BMP format (which are hidden in a header file), and that a third-party viewer or movie maker is required.

VIII. Final Comment

Most students, including underrepresented students, like the fractal very much. For example, some of my students have used it as background for talk slides, web pages, and even personalized credit cards. One student liked it so much that she gave me a jigsaw puzzle of the fractal as a thank-you gift.