**Assignment 2(OpenMP Tasking) Documentation**

Md Rafiqul Islam - 12123971

Technology used:

C++, OpenMP

How to run program:

1. Compile .cpp files:

* g++ a2-sequential.cpp -o a2-sequential.exe (sequential code)
* g++ -fopenmp a2-openmp.cpp -o a2-openmp.exe (for OpenMP)

1. Run program:

* ./a2-sequential.exe
* ./a2-openmp.exe

Code Structure:

The program consists of three different C++ files: a2-helpers.hpp, a2-sequential.cpp and a2-openmp.cpp.

* a2-helpers.hpp – This is header file which contains the defined data structures of the program. It has Image data structure, gradient data structure, and some functions called interpolate\_rgb\_color, colorize and get\_2d\_kernel. Here, interpolate\_rgb\_color sets the image color where it needs. Colorize function takes care of gradient color of the picture. And get\_2d\_kernel () function gives the kernel which one used for image filtering.
* a2-sequential.cpp – With the help of this c plus plus file we generate our Mandelbrot set and, we print image of it and filtered it. First, a set of random gradient values adjusted for our Mandelbrot algorithm. Here we have Mandelbrot\_kernel () function which check if the given point is a member of the Mandelbrot set or not, using z = z \* z + c this formula. If the point was a member of Mandelbrot set, then we colored then point with a color otherwise we used different color to identify the point. In our case, we just used maximum 2048 iterations.

Another fuction mendelbrot () is takes as parameter an image and ratio then it takes image’s height, width, and channels then for every pixel of image it calls Mandelbrot\_kernel () function. Then Mandelbrot\_kernel () checks if the given pixel is a member of Mandelbrot set or not. If it was a member then it colored the pixel in black otherwise different gradient color.

There is another function in this file called convolution\_2d. This function is used for applying Gaussian filter for this Mandelbrot set image. Note that here all works had been done in a single processor. That’s why this approach is slower.

* a2-openmp.cpp – This is our parallel processing

Tables:

|  |  |  |  |
| --- | --- | --- | --- |
| Threads | Time | Speedup | Efficiency |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Variant | Time | Speedup | Efficiency |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Speedup Graph and Description:

Fig-2: Image1 speed up curve.

Graph Description:

Discussions:

* Performance differences between task and task loop versions:
* Task granularity matters (small vs big tasks):
* Number of tasks generated:
* Differences in speedup:
* Interesting findings: