ACSE-6: OpenMP Group Assignment

Team Name: Team Inheritorplus

Date:17/02/2021

1. Code structure

1.1. Description

- **'Serial.h', 'Serial.cpp'**

This file contains the serial version of Conway's Game of Life. Functions to generate the output to .txt file and .png have also been added.

- **'Parallel.h', 'Parallel.cpp'**

Based on the serial version, a parallel version of Conway's Game of Life using OpenMP has been written here.

- **'stb_image.h', 'stb_image_write.h', 'bitmap_image.hpp'**

These files contain the code of libraries for generating image.

- **'Pixel.h', 'Pixel.cpp'**

The purpose of these files is to set the color of image using parameters.

- **'PNG.h', 'PNG.cpp', 'Image.h', 'Image.cpp'**

The purpose of these files is to create and save image.

- **'Interface.h','Interface.cpp'**

The purpose of these files is to generate an interface for users to implement the parallel version and compare it with the serial version. In the interface, users are able to generate or load grids of different size. The output images and .txt files are stored in ./png and ./txt separately.

1.2. Methodology of implementing the paralleling version

1) We store a square grid as a 1-dimension array, using column-major ordering, and

```
value = new bool[size_of_value];
```

also other 1-D array `neigh_value[]` , `row_index[]` and `col_index[]` to represent the number of live neighbours, row index and column index for a certain element with index **i** in the 1-D array.

2) The `add_neighbours` function in Parallel.cpp using omp parallel for

```
void Parallel::add_neighbours() {
#pragma omp parallel num_threads(threads) firstprivate(size, last, size_of_value, last_col_start, diag, row_index, col_index)
{#pragma omp for
        for (int id = 0; id < size_of_value; ++id) {
            if (value[id]) { if (....)
      #pragma omp atomic ++neigh_value[...];
      .....}}}
```

would traversed over live cells (value[id] == 1) in the 1-D array and add counts `atomically` for its 8 neighbours (in the periodic mesh).

3) The `do_iteration_parallel` function in Parallel.cpp would then traversed over the 1-D array `neigh_value[]` to see whether it's suitable for a cell to live in the next generation, it's the same logic as in the `ConwaysGame_Serial.cpp` but in parallel.

1.3. Strength and weakness

- **Strength**
  - The elements of the gird are traversed in a 1-D array, achieving faster calculation.
  - The input can be generated by our program or read from a .txt file.
  - An interface has been generated. In this way, users can input grid size, iterations, number of threads directly, which is very convenient.
- **Weakness**
  - Many `if` statement in the main algorithm, could do better use % operators.
  - Strict formatting on inputing the .txt files.
  - Printing takes too much time.

2. Input and output

2.1. Input

The input can be a random initial condition generated by our program or loaded from a .txt file.

2.2. Output

Table of output with max iteration of 100.(with printing time)

num. of threads/ grid size	100	500	1000	2000	10000
Serial	0.278966 s	6.55997 s	26.1442 s	106.263 s	2641.05 s
4	0.020556 s	0.371142 s	1.39001 s	5.61909 s	155.418 s
8	0.021774 s	0.338768 s	1.26496 s	5.12868 s	140.172 s
12	0.023730 s	0.332375 s	1.22448 s	4.98164 s	129.229 s
24	0.030101 s	0.331765 s	1.26222 s	4.99525 s	136.024 s
max acceleration times	13.6	19.8	21.4	21.3	20.4

Table of output with max iteration of 1000.(with printing time)

num. of threads/ grid size	100	500	1000	2000	10000
Serial	2.22705 s	55.1554 s	220.229 s	911.098 s	>7 hours
4	0.082084 s	0.802354 s	3.15873 s	12.4915 s	318.506 s
8	0.087977 s	0.655586 s	2.54007 s	9.97132 s	267.517 s
12	0.099249 s	0.604079 s	2.33833 s	9.42008 s	236.429 s
24	0.146722 s	0.645278 s	2.3476 s	9.37413 s	235.88 s
max acceleration times	27.1	91.3	94.1	97.2	>106.8

Table of parallel output with/without printing time.(Iteration:100, num. of threads:12)

grid size	100	500	1000	2000	10000
with printing time	0.023730	0.332375	1.22448	4.98164	129.229

without printing time	0.008286	0.031969	0.112073	0.460721	13.3509

Here are some key comments from the table:

- The time of acceleration increases as the grid size increases.
- When iteration increases, serial execution time increases proportionally but parallel increases slightly.
- Generally speaking, parallel execution time decreases when the number of cores increases.
- Printing takes about 90% of the execution time.

3. Execution

To execute the program, please refer to the guide in 'ReadMe.md'.

4. Other information

- Github Link: https://github.com/acse-2020/group-project-inheritorplus
- Compiler(s) used: g++-10
- OpenMP version: 5.0
- Operating system: macOS Catalina 10.15.6
- Number of logical processors: 6
- Members: Jin Yu, Xuyuan Chen, Yuchen Wang
- Breakdown of work:

Jin Yu: serial and parallel algorithm coding, interface coding, 'ReadMe.md' writing

Xuyuan Chen: image generation coding, interface coding

Yuchen Wang: report writing

5. References

Deborah Pelacani Cruz & Hugo Coussens (2019), IMAGE FILTER SOFTWARE (ACSE-5 Homework-2 Example solution).