Parallel Assignment 1

Wang Chenyu & Cai Xuemeng

Implementation assumptions

We assume the overhead of creating threads is not too big

Parallel Strategy

OpenMP: I meanly paralleled all the double for loops that involved getNextState() function and setValueAt() function, paralleling the getNextState() can significantly speedup the running time because it takes a relatively long time to execute, when the world size increases, it occupies most of the process running time, dividing them to different threads will gain a nearly linear speed up if thread creation overhead is small. What is more, there is no synchronisation needed as each thread will read only write to their own slots. There is no case that 2 threads are writing the same slot, and because that we made a copy of the world at the very beginning, we no need to worry about one thread is reading one slot while another thread is updating the slot.

PThread: I think the most heavily computed part is the part to calculate the next states of each grid so I mainly partition that part into several threads. It is a task parallelism since the task to calculate each grid's next state as a single task and the data is shared among threads. The method I use is master-worker. The main thread as a master will partition the work load and send out to each threat which is the worker.

Implementation detail (OP)

OpenMP: There is no need for us to explicitly define which variable need to be set as shared variable and which variable need to be set as private variable. If we didn't pre-define the two counters for the nested for loop (row and column), it will be automatically defined as private variable and all other variable which are pre-defined before the #pragma omp declaration will all be shared variables. That is exactly what we want to achieve. But do remember to declare the num_threads for this program.

PThread: The most important part is to know how to assign each task to each thread properly. I used one strategy to assign the tasks most equally to each thread by calculating (row * nCol + col) % nThread to decide the thread id that one task should go. The method for the master and worker to communicate is an array of a customised data type which is called ThreadData. This array contains all the work that the master assigned to each thread and each thread will send back the results by changing this array's data.

Reproduce Guidance

make build
./<thread_file_name> ./testcases/inputs/<input file> ./testcases/outputs/<output_file> <thread num>

E.g

```
./goi_threads ./testcases/inputs/input4.in ./testcases/outputs/output4.out 16
```

```
./goi_omp ./testcases/inputs/input2.in ./testcases/outputs/output2.out 8
```

Execution time and speedup measurements

ps: all tests are being ran on soctf-pdc-005 (Single Xeon Silver 4114)