Final Project

Introduction to Parallel Computing

NTHU ISA | 110065504 | 葉之晴

2. ALGORITHM

3. RESULT

1. INTRODUCTION

4. CONCLUSION

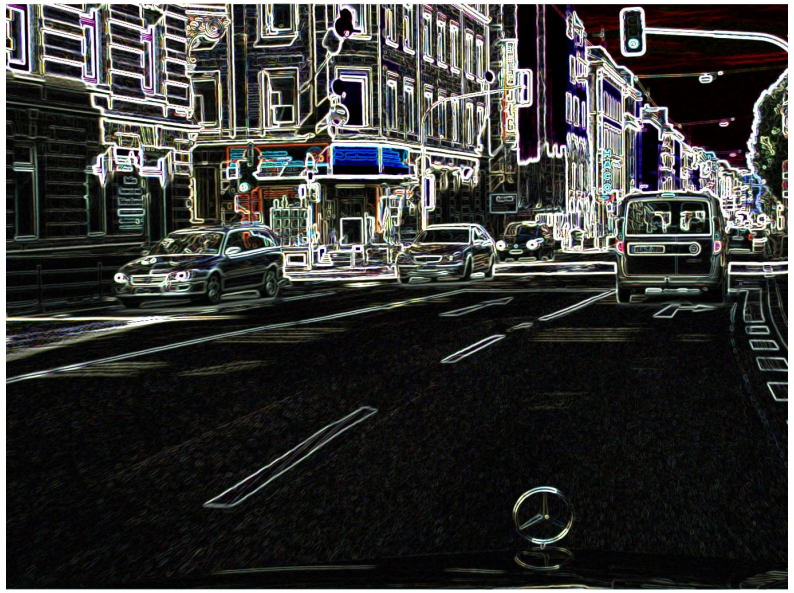
1. INTRODUCTION

Conway's Game of Life

MOTIVATION

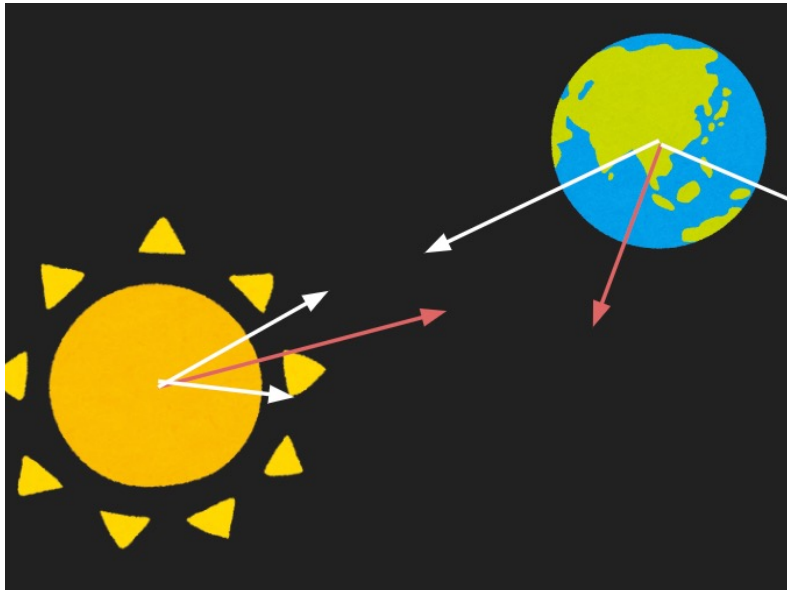
HW3: Sobel

Key: Pixel



HW5: N-body Simulation

Key: Simulation steps(200000)

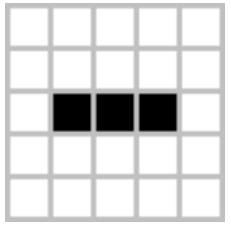


Conway's Game of Life

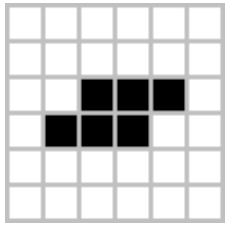
MOTIVATION

I implemented a parallel Conway's Game of Life with dynamic work allocation on both CUDA and OpenMP.

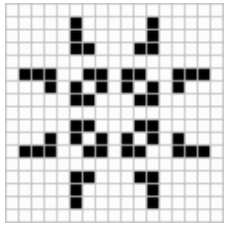
Conway's Game of Life



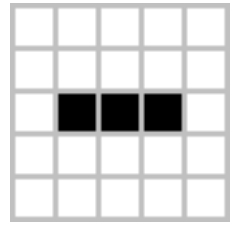
INTRODUCTION



- John Conway came up with the Game of Life in 1970.
- The game demonstrates the fact that some simple local rules can lead to interesting large-scale life behavior.
- The game is played in a two dimensional grid N x N, made of cells, that can be either alive, or dead.
- The game does not have any players, and each cell has at most 8 neighbors, that determine its state in the next generation.

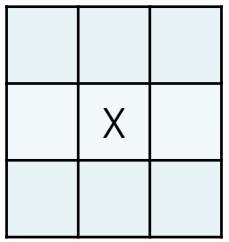


Conway's Game of Life



INTRODUCTION

- The reformation of the grid from generation to generation is done simultaneously, meaning that each state in the next generation depends exclusively in the state of the cell and its neighbors.



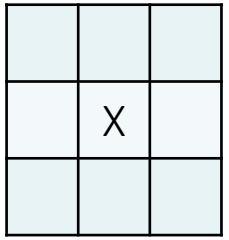
Conway's Game of Life

Rule

Cell state

Number of living neighbors

	<2	2	3	>3
Alive	Die	Continue to live	Continue to live	Dead
Dead	Remain dead	Remain dead	Come to life	Remain dead



Conway's Game of Life

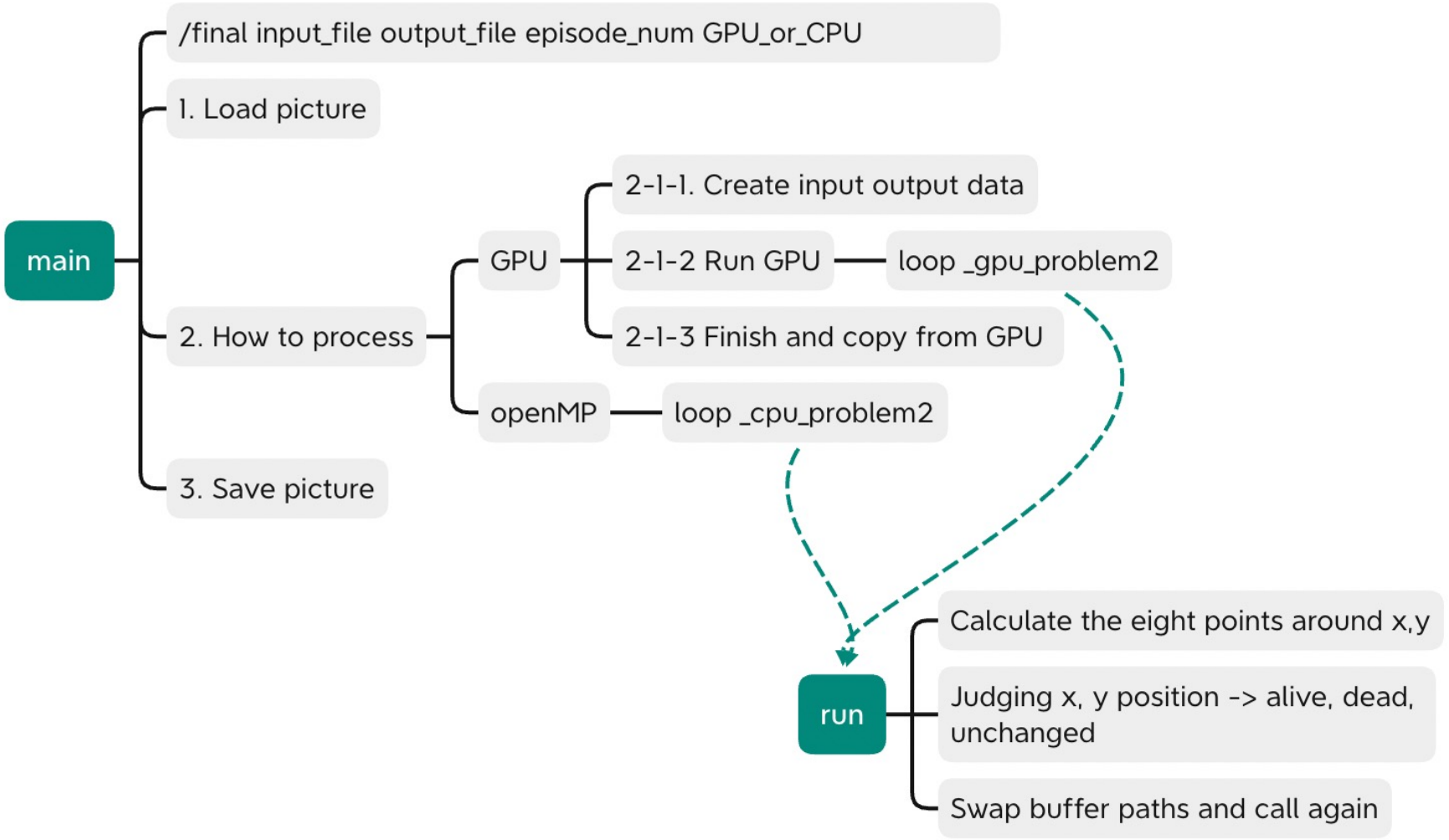
Rule

Algorithm 1 ALGORITHM NEXTGENERATION(*cell*)

```
num_neighbors ← cell.COUNT_NUM_NEIGHBORS()
if cell.ISALIVE() then
  if num_neighbors < 2 or num_neighbors > 3 then
    cell.setState(DEAD)
  end if
else
  if num_neighbors = 3 then
    cell.setState(ALIVE)
  end if
end if
```

2. ALGORITHM

Program Structure

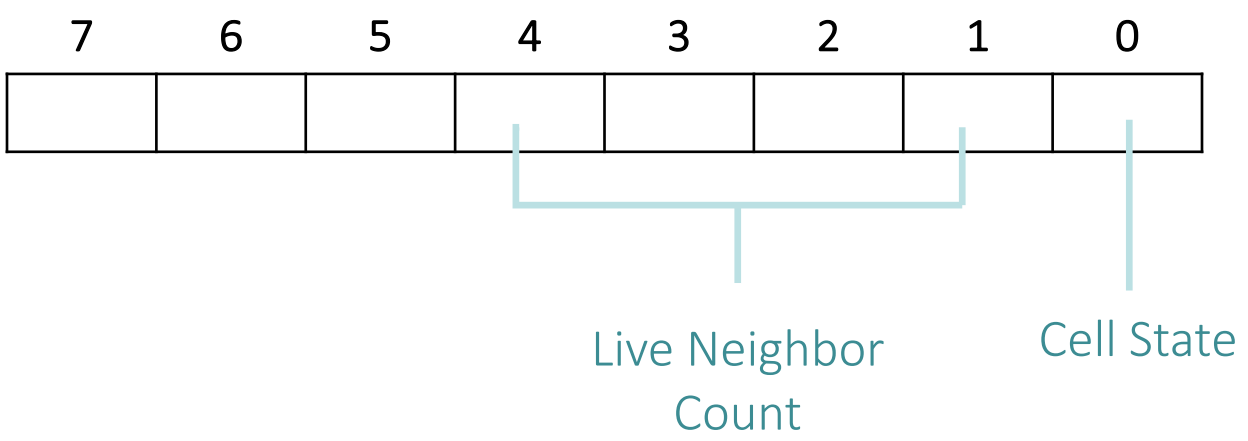


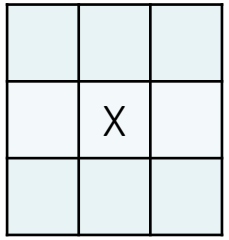
Program Structure

1. The difference in operation between opemMP and Cuda after running 1000, 2000, …, 5000 times respectively.
2. Give a grayscale (1byte) black and white image, after N life cycles, output .png and view the result.
3. In order to reduce the time of copying data, a **rolling storing area** will be established. The operation function will be called twice during processing, and the interleaving drawing is a loop-seeking. (Ex. After the data of A finish to calculate, the result is stored in B. Then, after the data of B finish to calculate, the result is stored in A.)

Pruning

- Data structure -> char





- Multiple bounding boxes.
- Simple bounding boxes.

-> if (!cell_state[x][y] && alive_neighbor[x][y] == 0)

Parallelization Goal

- Reduction of idle time
- Reduction of unnecessary computations
- Parallelization of a great portion of the program
- Use of datatypes

- Rule (3*3)
- Alive State (512*512)

OpenMP vs. Cuda

- Rule (3*3)
- Alive State (512*512)

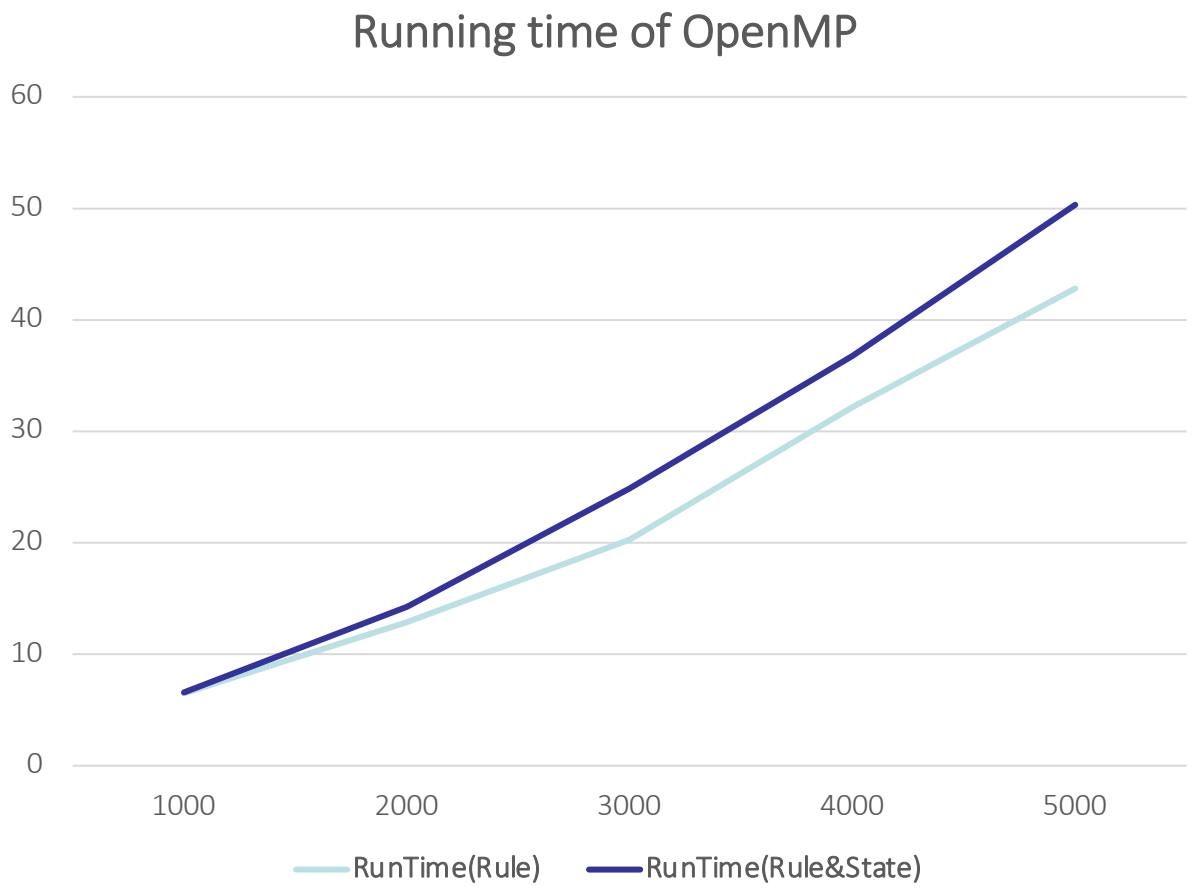
	OpenMP	Cuda
RunTime	6.58	9.25

- Alive State -> Need to synchronize
- Only parallelize the action of doing the rule

OpenMP

#pragma omp parallel for

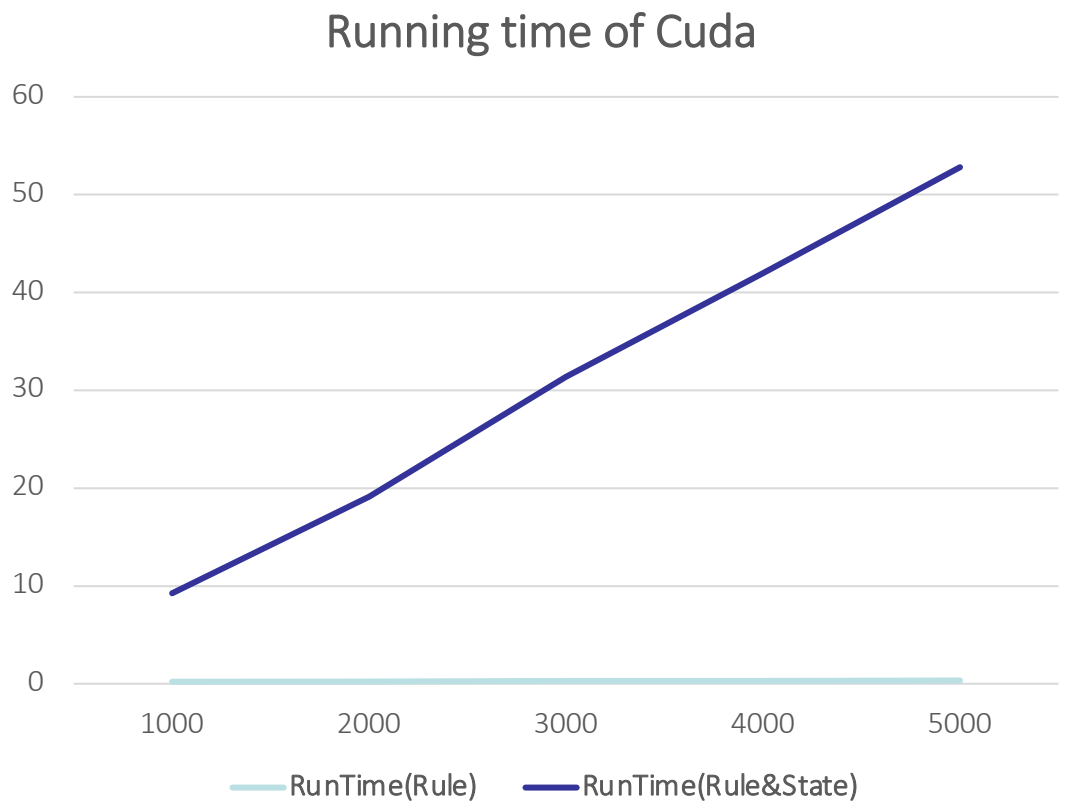
- Rule (3*3)
- Alive State (512*512)



Cuda

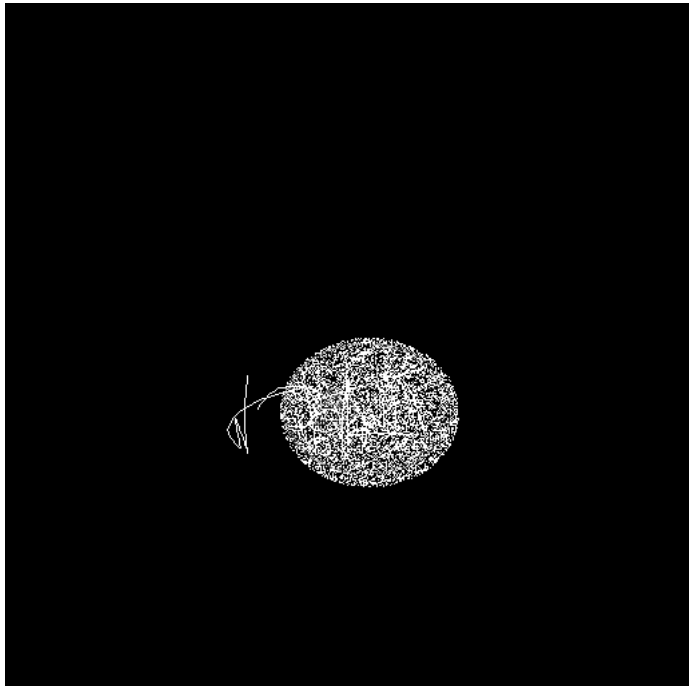
x=blockIdx.y*blockDim.y + threadIdx.x;
y=blockIdx.x*blockDim.x + threadIdx.y;
-> thread blocks: 8*8*1
-> each thread block: 64 threads

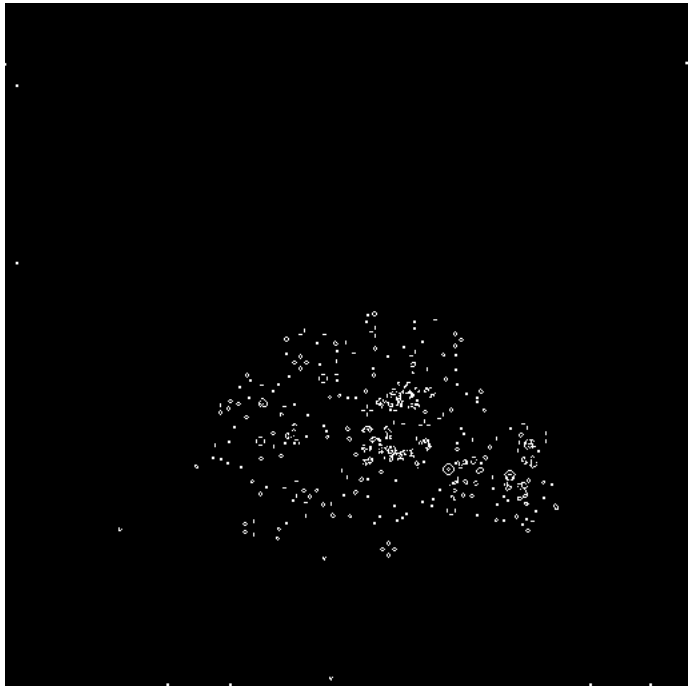
- Rule (3*3)
- Alive State (512*512)



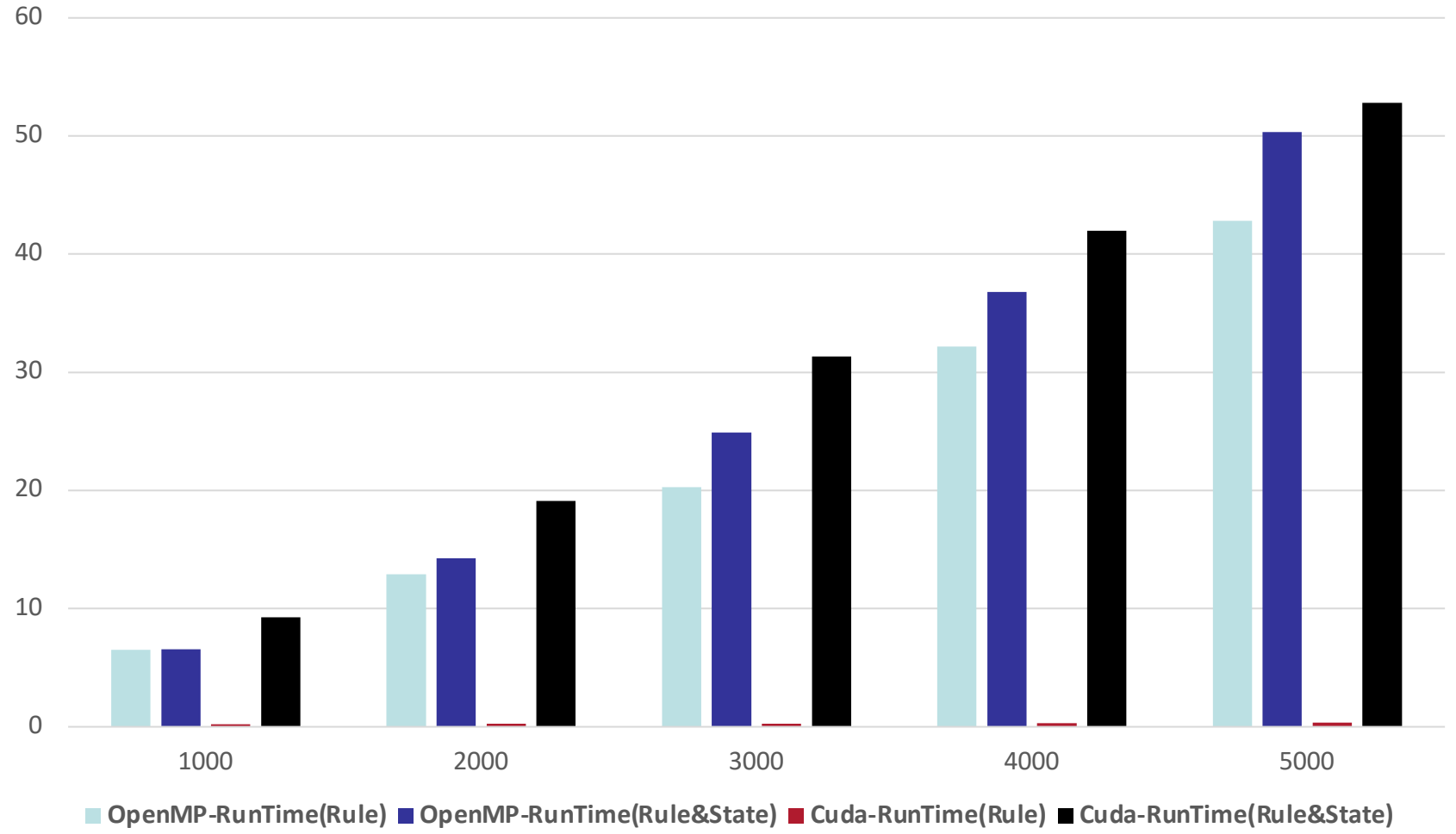
3. RESULT

RESULT





RESULT


60
50
40
30
20
10
0
1000
2000
3000
4000
5000
OpenMP-RunTime(Rule)
OpenMP-RunTime(Rule&State)
Cuda-RunTime(Rule)
Cuda-RunTime(Rule&State)

4. CONCLUSION

Conclusion

	Rule & Alive Neighbor	Rule
Fast	OpenMP	Cuda
Time	1.04x	128.81x

Q & A

Q: In the class last week, the professor and the TA recommended that adding the stream parallelism function would reduce the computing time. Here I would explain this problem in detail.

A: When I add the stream to parallelize, the result shows that the computing time is much slower, almost equal to the result when I did all the parallels on page 15, about ten seconds. I think the reason is still the same as on page 15 because the alive state needs to wait for everyone (512*512) to finish before executing the next time. In summary, that the results of this time will be used next time is the key point.

Q & A

Because this question was not explained clearly in class, I would like to explain all of the questions in detail here.

1. I put the data on the GPU and did not move the data back to the CPU. I only read it after trying N times.
2. About the above-mentioned reason, "the results of this time will be used next time", it is like saying that a group of people go travel in a tourist bus, and they will drive only when all of them arrive, but after arriving at the destination, they can go and play on their own.
3. Therefore, after removing the parallelism of the alive state, the following two times can be solved.
    1) thread synchronization
    2) The number of for judgments

THANK YOU

葉之晴