Speedup Analysis of Conway's Game of Life on the Prallella Board

Conrad Berlinguette
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
cberl060@uottawa.ca

Alexandre Pante
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
aplan084@uottawa.ca

***Abstract*—Conway's Game of Life is a fun and interesting way of visually demonstrating processor parallelism at work. Using the Parallella as a platform to do so, we demonstrate and discuss speedup metric related to the game.**

Keywords— Parallella, Conway's game of life, parallelism, Speedup

I. INTRODUCTION

Despite being the size of a credit card, the Parallella board requires a lot of knowledge and equipment to get up and running. We required the physical board, power transformer, Ethernet cable to connect the board to a router, a MicroSD card, and a cooling fan. In terms of software, the bare bones we used included PuTTY and Notepad++. Using these tools, we implemented Conway's Game of Life using the MIT code repository.

II. OVERVIEW

A. The Game

In 1970, British mathematician John Conway devised a game concerning cellular automaton. The game is a discrete model studied in computer science modeling, where an n by n grid of cells is supplied. Each cell is binary: 'x' (dead) or 'o' (alive). For each cell, a decision is made mathematically on whether it lives, dies, or multiples.

The algorithm can run for an arbitrary number of iterations, and each new iteration is calculated from the previous one. Each value is calculated only by using its neighbor's values, and it is therefore very parallelizable. Depending on the initial conditions, the cells form various patterns throughout the course of the game. The rules are follows:

- For a space that is 'populated':
  - Each cell with one or no neighbors dies, as if by solitude.
  - Each cell with four or more neighbors dies, as if by overpopulation.
  - Each cell with two or three neighbors survives.
- For a space that is 'empty' or 'unpopulated':
  - Each cell with three neighbors becomes populated

B. Implmentation

Noémien Kocher wrote code for a variation of the game to run on the Parallella in 2016. He set a fixed grid size of 4x4, which equates to 16 cells, meaning one per processor. The user is permitted to set the number of iterations/state changes they wish to see. We added a timer feature to calculate the time spent between states, to give us a better understanding of how fast the calculations were performed.

III. Findings

On average, each iteration took approximately 0.2 milliseconds. Most of that processing time would have been compounded by the number of cells if run in a serial fashion, although we lacked the control single out only one core for processing. Because we do not have a precise baseline, our speedup time is more of an assumption.

The 0.2 milliseconds represents the time it takes for a Next State to be generated from a Previous State. We know that a cell's Next State is independent of any other cell's Next State, and therefore its Next State calculation can be parallelized. In our case, each cell can be independently calculated by its own dedicated processor. This means there is initialization overhead, and likely some minor iteration or printing overhead, but the bulk of the program (i.e. calculating the cell state at each iteration) is completely parallelized. This would lead us to an assumption that the speedup of a 4x4 grid, from a single core to 16 cores, would be close to a value of 16. This result should stand more or less regardless of the number of iterations, the greater the number, the closer to a speedup of 16.

We also did some research on the game and what other people had done as implementations. It appears our estimations were not far off, as Trygve Aaberge wrote a report and found that the Epiphany was 13 to 16 times faster than the main single core processor. In his case, the grid size was significantly larger (180x180), and his results for total program runtime, in seconds, were as follows.

Frames	Epiphany	Xilinx, array
100	0.223	3.222
2000	4.429	59.506
10000	22.106	291.853

Figure 1: Trygve Aaberge's Report

IV. Conclusion

Conway's Game of Life provided a fun opportunity to visually represent parallel processing. Because of the inherit independence of the game's elements, we could easily implement parallelization for the bulk of the program. Our findings proved similar to other field research, and we are comfortable in our implementation and assumptions to say that the speedup offered by the Parallella board is close to 16.

References

[1] Noémien Kocher, Conway's Game of Life GitHub repository, published 2016, accessed November 2018. https://github.com/parallella/parallella-examples/tree/master/game-of-life

[2] Trygve Aaberge, "Analyzing the Performance of the Epiphany Processor", published August 2014, accessed November 2018. https://daim.idi.ntnu.no/masteroppgaver/011/11745/masteroppgave.pdf