Parallel Programming Exercise 6 – 13

Author:	李子筠(b06901145@ntu.edu.tw)
Student ID	b06901145
Department	Electrical Engineering

(If you and your team member contribute equally, you can use (co-first author), after each name.)

1 Problem and Proposed Approach

(Brief your problem, and give your idea or concept of how you design your program.)

題目要求我們模擬 John Conway's game of life，從檔案內讀進 grid 大小以及初始 state，跑指定的 j 個 iteration，每 k 個 iteration 印出目前的 state。

讀檔的部份我照上課的方法，用 row striped 的方式將 grid 切給每個 process，讓第 p-1 個 process 負責 I/O，再用 Send/Recv 傳給其他 process。

接著是更新 state 的部份，因為需要知道周圍 8 個 cell 的 state，所以每個 process 都先把第一列和最後一列傳給鄰居。更新 state 的方法就用 sliding window 去取 3x3 的 9 個 cell 中 (在邊界是 6 個) cell 的 live cell 個數，再扣掉自己的 state 來得到周圍的 live cell 總數。

為了 inplace 更新 grid，我將 cell 存在 8 個 bit 中，前 4 bit 存更新後的 state，後 4 bit 存更新前的 state。用上面 window 的方式更新完 grid 之後就用 shift 把新的 state 放到後 4 bit。

印出 state 的方式是用 Send/Recv 讓所有 process 輪流傳 grid 給 process 0，process 0 再按照順序印出。這樣就只有 process 0 需要 2*n/p 的空間來存 grid，不像 gather 每次傳輸後資料都會變兩倍。

2 Theoretical Analysis Mode

(Try to give the time complexity of the algorithm, and analyze your program with iso-efficiency metrics)

M: row 個數

N: column 個數

p: process 個數

i: 總共跑幾個 iteration

j: 每幾個 iteration 印出 state

χ: 更新一個 cell 所需的時間

λ: process 間傳訊息所需時間

令 j = 0

Sequential time complexity: χMNi

Parallel:

I/O 時 process p-1 做 MPI_Send: $(p-1)(\lambda + \frac{MN}{p\beta})$

每個 iteration MPI_Send/MPI_Recv 第一個和最後一個 row: $(\lambda + \frac{2N}{\beta})i$

每個 iteration update 自己的 row: $(\chi\frac{MN}{p})i$

Parallel time complexity: $$(p-1)(\lambda + \frac{MN}{p\beta}) + \left[(\lambda + \frac{2N}{\beta}) + \chi\frac{MN}{p}\right] i$$

3 Performance Benchmark

(Give your idea or concept of how you design your program.)

Table 1. The execution time

Processors	1	2	3	4	5	6	7	8
Real execution time	1.811	0.908	0.621	0.468	0.374	0.313	0.270	0.235
Estimate execution time	1.812	0.906	0.604	0.453	0.363	0.302	0.259	0.227
Speedup	x	1.99	2.91	3.87	4.84	5.77	6.70	7.67
Karp-flatt metrics	x	0.002	0.014	0.011	0.008	0.007	0.007	0.006

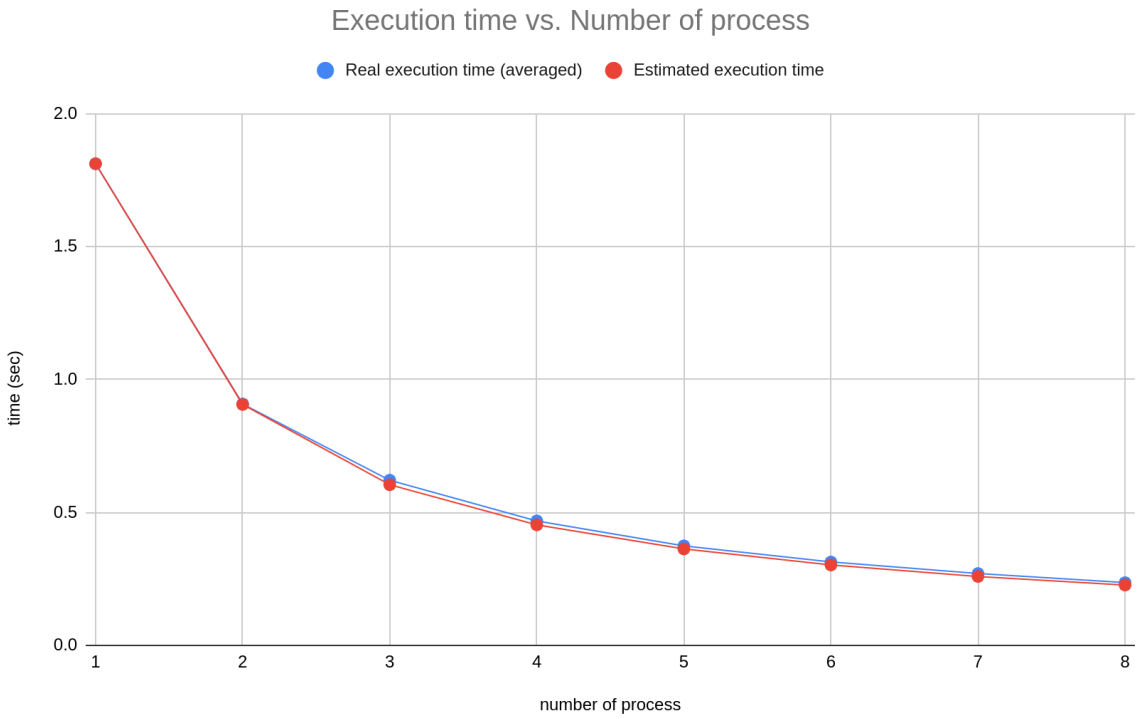


Figure 1. The performance diagram

4 Conclusion and Discussion

(Discuss the following issues of your program

1. What is the speedup respect to the number of processors used?
2. How can you improve your program further more
3. How does the communication and cache affect the performance of your program?
4. How does the Karp-Flatt metrics and Iso-efficiency metrics reveal?

)

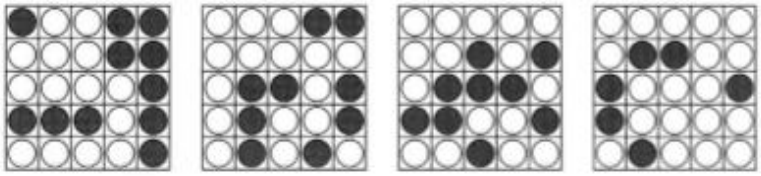
1. speedup 隨著 p 上升而上升，增加的越來越少。
2. I/O 的部份可以用 MPI_File 輸出，而不是將 row 都傳給 process 1 輸出。
   可以將 grid 切成 block，如果 block 周圍的 cell 沒有變化，而且自己前一次也沒有變化，就可以不用更新它。
3. 每個 iteration 都有 Send/Recv，不過傳送的資料相對整個 grid 來說小很多，所以花的時間相對 sequential update 小很多。
4. Karp-Flatt metrics 大致隨著 p 上升而下降，代表 sequential 佔的比例不小

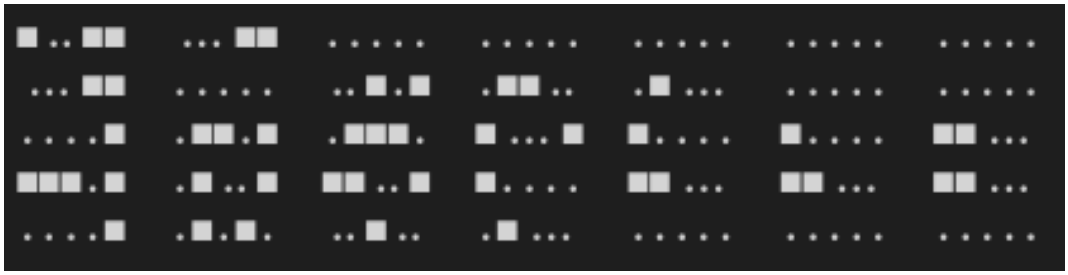
Appendix(optional):

(If something else you want to append in this file, like picture of life game)

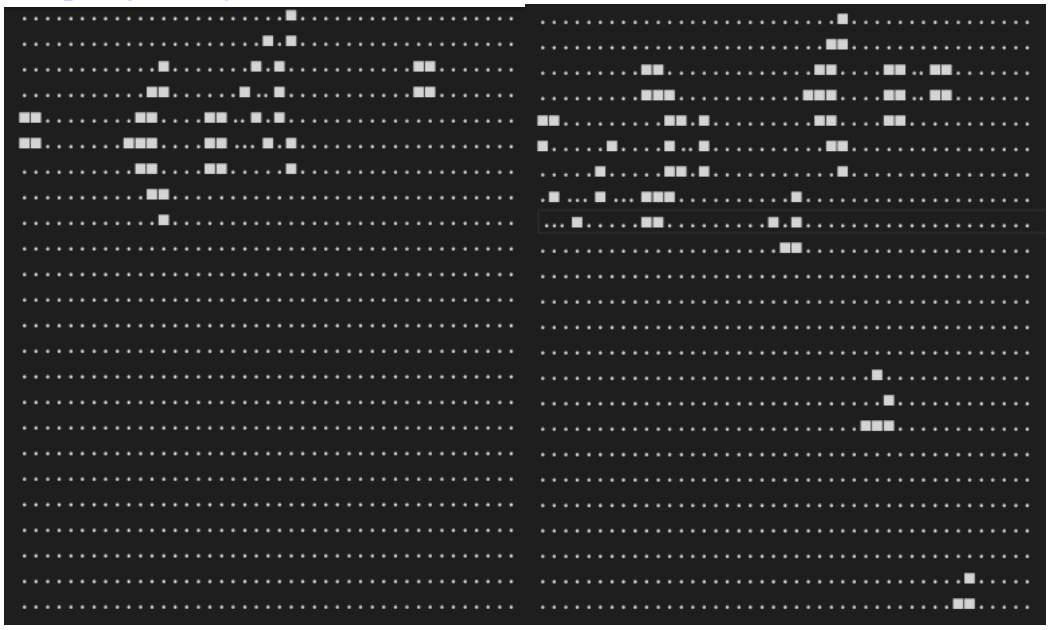
(replace 0 by '.' and 1 by '■' for readability)

Example





[Gosper glider gun]



[Period-20 glider gun]

80 iterations