HW2 – Report

1. **Name:** 徐嘉駿, **Institute:** 資應所, **Student ID:** 107065528

2. Implement

- **Pthread:**

  理念: 切高度分給 threads 做 dynamic 計算。

  (1) 設置一全域變數 *current_height=0*。

  (2) 啟動 threads，每個 thread 取得 lock *current_height* 並加 1 後再 unlock。

  (3) 利用 *pthread_join* 等待全部 threads 都完成任務後再畫圖。

- **Hybrid:**

  理念: 切高度分給 processes，process 再利用 threads 做 dynamic 計算。

  (1) 每個 process 宣告變數 *local_start* 為自己的 rank ID。

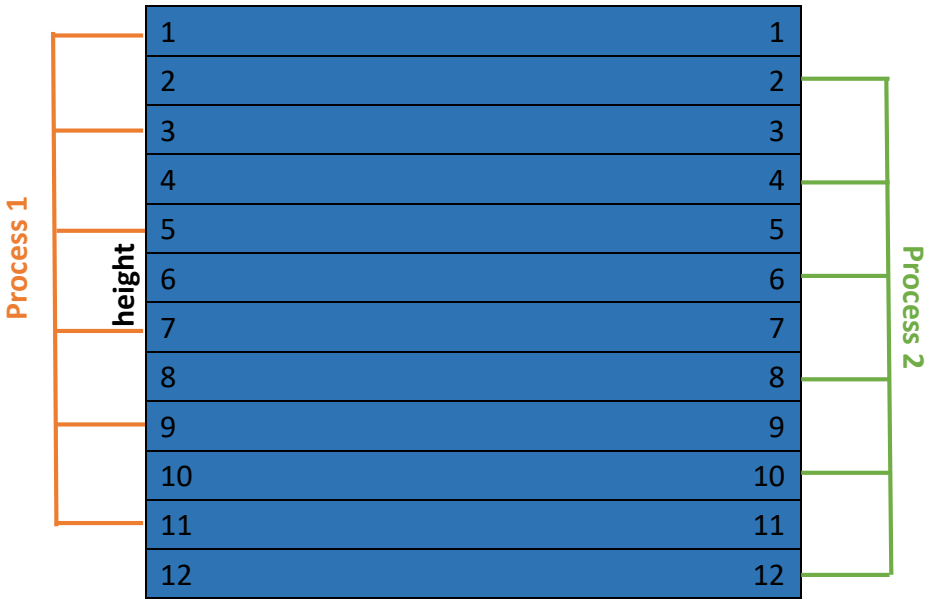
  (2) 每個 process 宣告變數 *image* 當作 local array，只有 rank = 0 的 process 再多宣告一變數 *global_image* 當作最後收集資料(畫圖用)的 array。

  (3) 開始計算 mandelbrot set 之前，每個 process 開啟自己最多能啟動的 thread 數量。

  (4) 切 height，每個 process 利用 *local_start*(自己的 rank ID)當作 height index 開始處理每條 width，每次處理完後加上 process height。

  如圖:

  假設有兩個 process 可以使用，每個 process 要處理的部分(橘色及綠色)。



(5) 每個 process 啟動 threads 幫忙計算自己所分到的部分。
(6) 利用 MPI_Barrier 等待全部 process 完成各自的任務。
(7) Process rank = 0 利用 *MPI_Reduce* 來彙整全部 process 的結果(存入 *global_image*)，並畫圖。

3. Experiment & Analysis

- **Methodology:**

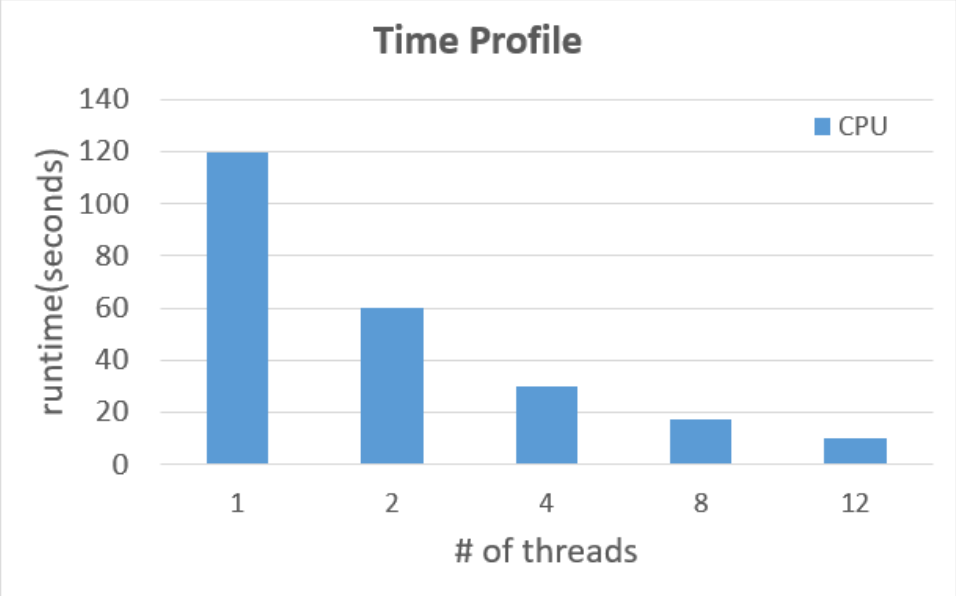
Performance Metric:

(1) **Computing time:** 有利用到 cpu 動作的時間都算在內，例: 計算 mandelbrot sort。

- **Time Profile & Speedup Factor:**

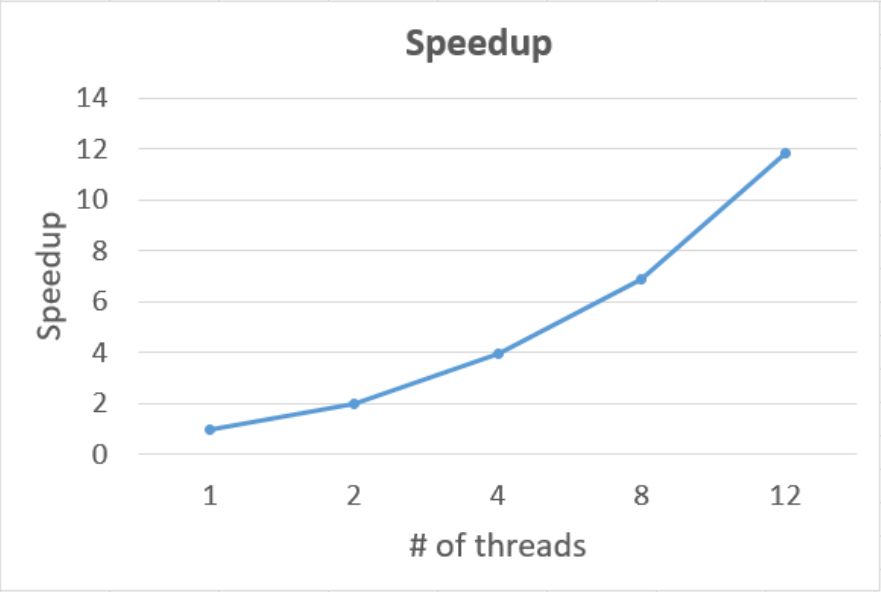
使用 slow04 做 measurement

(1) Pthread:

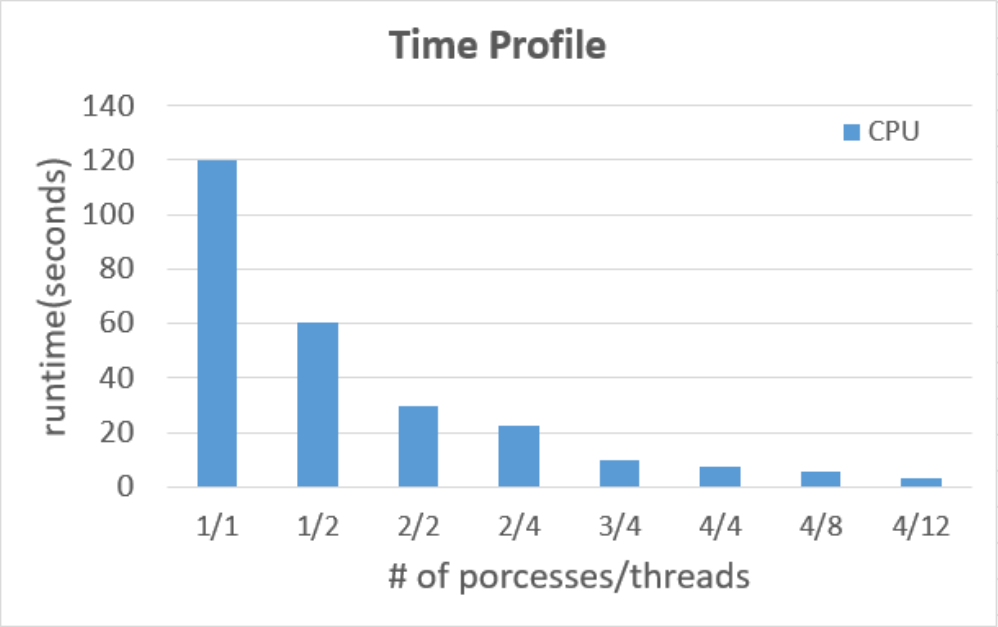


數值:

thread	CPU	Speedup
1	119.740211	1
2	60.053502	1.99389223
4	29.994899	3.992019143
8	17.342095	6.904598954
12	10.139797	11.80893572

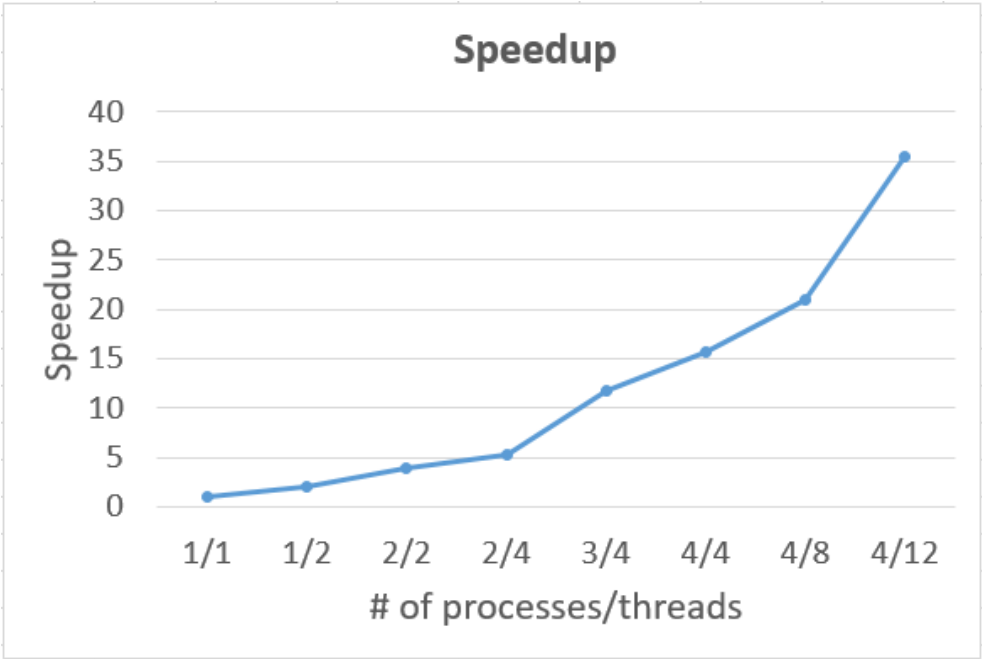


(2) Hybrid

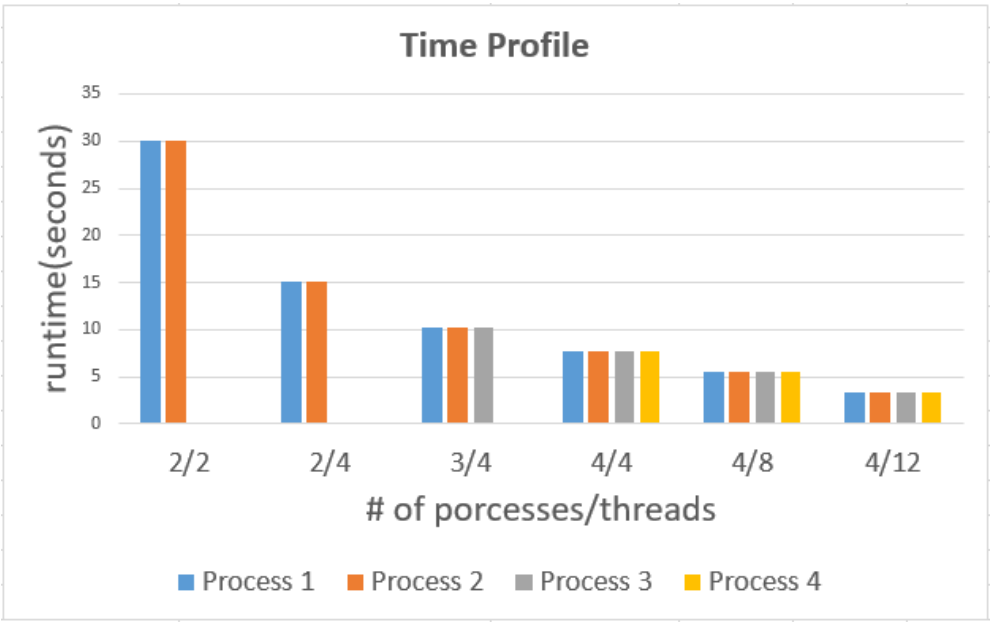


數值:

process	threads	CPU	Speedup
1	1	119.729254	1
1	2	60.308625	1.985275804
2	2	29.998008	3.991240152
2	4	22.557209	5.307804436
3	4	10.141085	11.80635543
4	4	7.676708	15.59643196
4	8	5.699719	21.00616785
4	12	3.380091	35.42190255



Load Balance



- **Discussion:**
  - **Scalability**

(1) **Strong Scaling:** 不論 Pthread 或者 Hybrid 版本，由圖可知，隨著 process 數上升，總體的數字量不變，每個 process 需要處理的量越來越少，parrallel 處理下來，computing time 也因此不斷減少。

(2) **Weak Scaling:** 拿 case *slow04* 為例，執行原本的 width & height 和 width, height 減少 2 倍，process 量減少 4 倍，由以下圖測試畫面可知，完成時間也是能幾乎達到一樣，也有接近 weak scaling 的概念。

PThread:

```
[pp19s96@apollo31 hw2]$ srun -n1 -c12 ./hw2a out.png 17326507 -0.550616469161878
3 -0.5506164628264113 0.6273445437118131 0.6273445403522527 1920 1080
computing time: 10.125009
[pp19s96@apollo31 hw2]$ srun -n1 -c3 ./hw2a out.png 17326507 -0.5506164691618783
 -0.5506164628264113 0.6273445437118131 0.6273445403522527 960 540
computing time: 10.079265
```

Hybrid:

```
[pp19s96@apollo31 hw2]$ srun -n4 -c6 ./hw2b out.png 17326507 -0.5506164691618783
 -0.5506164628264113 0.6273445437118131 0.6273445403522527 1920 1080
computing time: 5.174479
[pp19s96@apollo31 hw2]$ srun -n2 -c3 ./hw2b out.png 17326507 -0.5506164691618783
 -0.5506164628264113 0.6273445437118131 0.6273445403522527 960 540
computing time: 5.066160
```

■ Load Balance

(1) **Pthread:** 讓每個 thread 每次計算一個 width 量的點，做完後利用變數 *current_height* 再取尚未計算的一個 width 量的點，來達到 dynamic 分配。利用此方式讓每個 thread 都有算到比較困難的地方(點)來達到 load balance。

(2) **Hybrid:** 讓每個 process 都有被分配到簡單及困難的 height(一個 width 量的點)，每個 process 再加上 threads 幫忙加速計算，利用此種方法盡可能達到 load balance。

4. Conclusion

這次的作業讓我知道及如何使用 Pthread 及 OpenMPI+OpenMP。這次一開始寫時，並沒有遇到什麼大 bug，但寫出第一版 code 時，發現跑得很慢，甚至有時候還比 sequential code 慢。最困難的應該是該如何盡量讓每個 process 或 thread 處理相同的計算量，來達到 load balance，減少整體 completion time。