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)。

**width**

**Process 2**

**Process 1**

**height**

|  |
| --- |
| 1 1 |
| 2 2 |
| 3 3 |
| 4 4 |
| 5 5 |
| 6 6 |
| 7 7 |
| 8 8 |
| 9 9 |
| 10 10 |
| 11 11 |
| 12 12 |

* 1. 每個process啟動threads幫忙計算自己所分到的部分。
  2. 利用MPI\_Barrier等待全部process完成各自的任務。
  3. Process rank = 0 利用*MPI\_Reduce*來彙整全部process的結果(存入*global\_image*)，並畫圖。

1. **Experiment & Analysis**

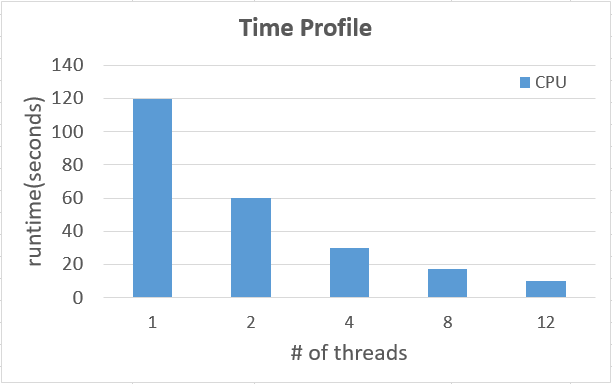
* **Methodology:**

**Performance Metric:**

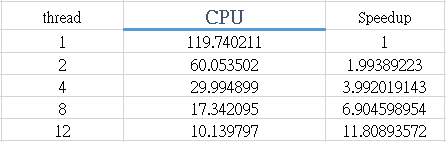
* 1. **Computing time:** 有利用到cpu動作的時間都算在內，例: 計算mandelbrot sort。
* **Time Profile & Speedup Factor:**

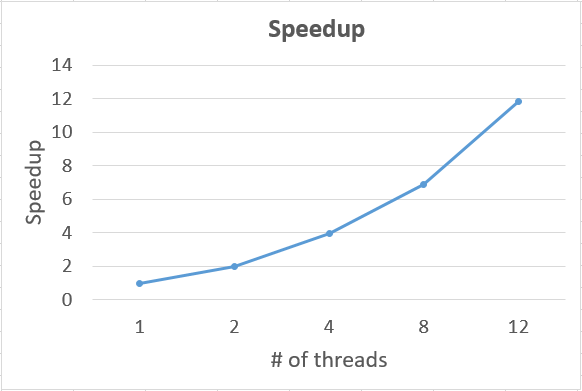
*使用slow04做measurement*

1. Pthread:

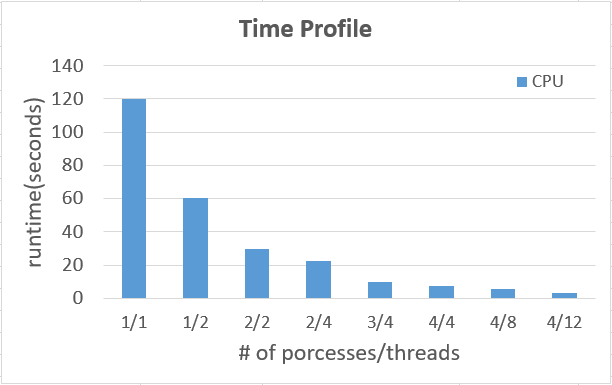


數值:

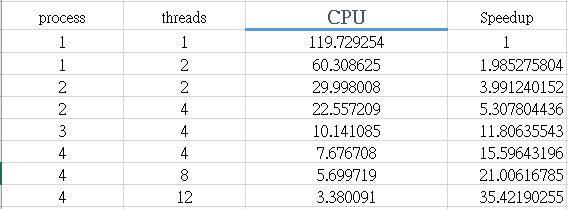


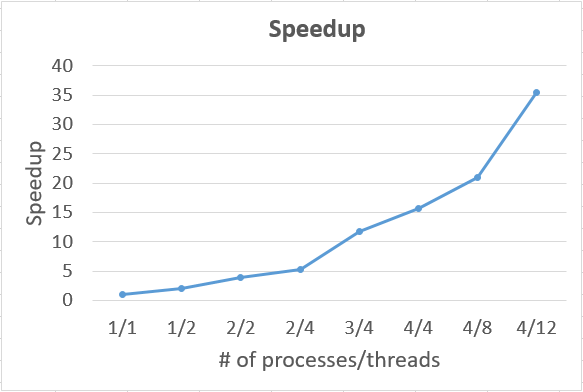


1. Hybrid

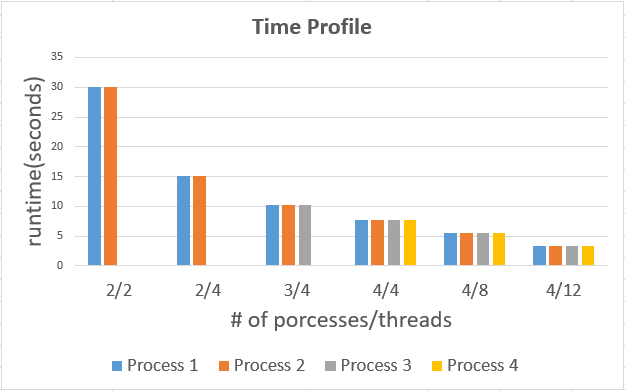


數值:





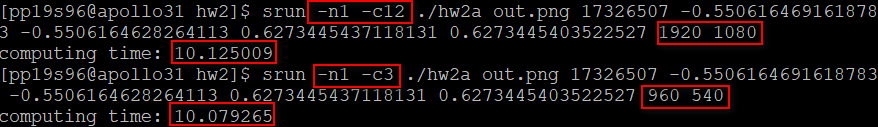
Load Balance



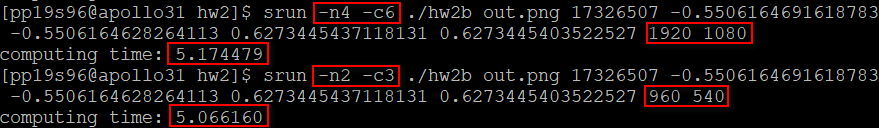
* **Discussion:**
  + **Scalability**

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:



Hybrid:



* **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。
3. **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。