# HW3：All-Pairs Shortest Path（CPU）

**Implementation：**

I. Which algorithm do you choose？

我選擇使用 Floyd-Warshall 演算法加上 Pthread 完成這次作業，由於考慮過其他常見的 APSP 演算法，發現其實 Floyd-Warshall 算是蠻快的方法，尤其在 edge 數量很多的情況下跟其他演算法相比更為快速。

II. What is the time complexity of your algorithm, in terms of number of vertices V, number of edges E, number of CPUs P？

一般而言 Floyd-Warshall 的複雜度為 O（$V^3$），但是從平行的角度來看複雜度為 O（$\frac{V^3}{P}$），符合 Cost Optimal algorithm。

III. How did you design & generate your testcase?

我的想法是讓 E 的數目越大越好，因為這樣 input 會多花一點時間，然後如果有人用 Dijkstra's algorithm 去實作(用 binary heap)，這樣複雜度為 O(VElgV)，所以當 E 越大，就會使得複雜度大於 O（$V^3$），大大減低速度，至於裡面的數字，我盡量數字越小的起始點會有越大的值，這樣透過 floydwarshall 的話更新 graph 的次數應該會增加，因而增加時間，最後透過 f.write()把產生的 binary file 生出來。

IV. Other efforts you've made in your program.

(1)

我的方法是將每一個一個 row 分給 threads，起出分的方式是每個 thread 都拿連續的 row，像是 0~10 列這樣拿，這種方法會造成當所有 thread 在讀寫矩陣的資料時，可能來回移動的距離會過大，因而降低效能，所以我將 thread 拿 row 的方法改為每 CPU core 列數拿一次，這種方法加快了許多時間。

(2)

對於讀取 binary file 裡的資料，起初使用 f.read()去讀，後來改成使用 mmap 去讀資料，如此可以加快讀取資料的速度，因為中間不需要經過 buffer，大概快了 5 秒左右。

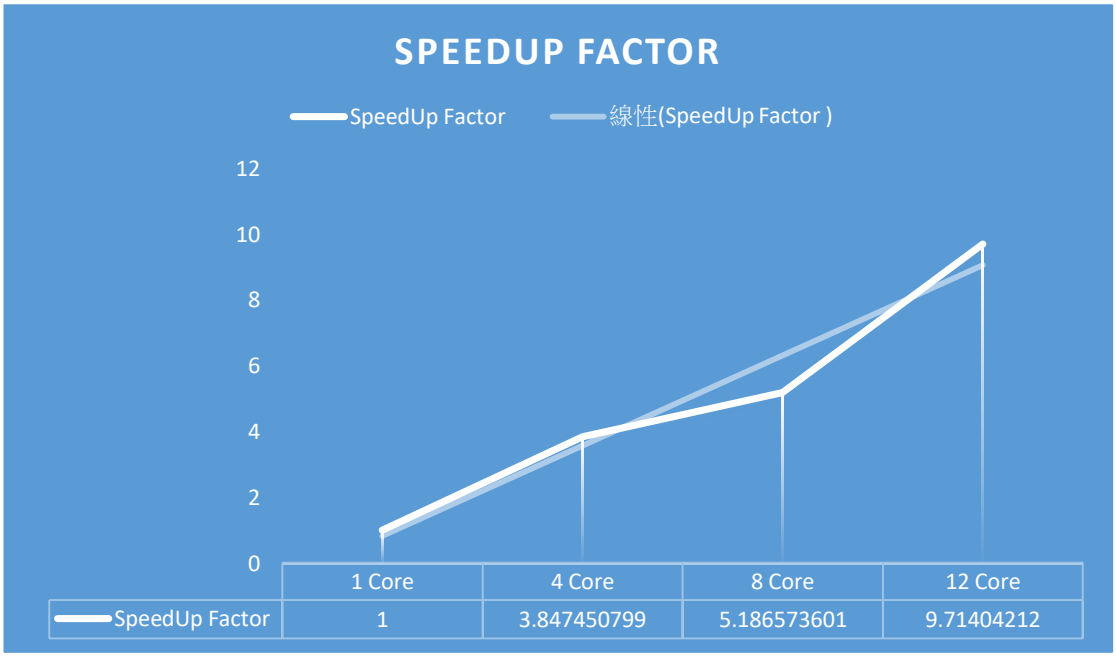
(3)

對於寫入 binary file，我也盡量減少 call f.write()的次數，以期許減少時間，這似乎也有些微的效果。

**Experiment & Analysis：**

這裡的分析是取 hw3-judge 中的 c21.1 測試，時間也保證夠長，使得結果有意義。

(b) Strong scalability：

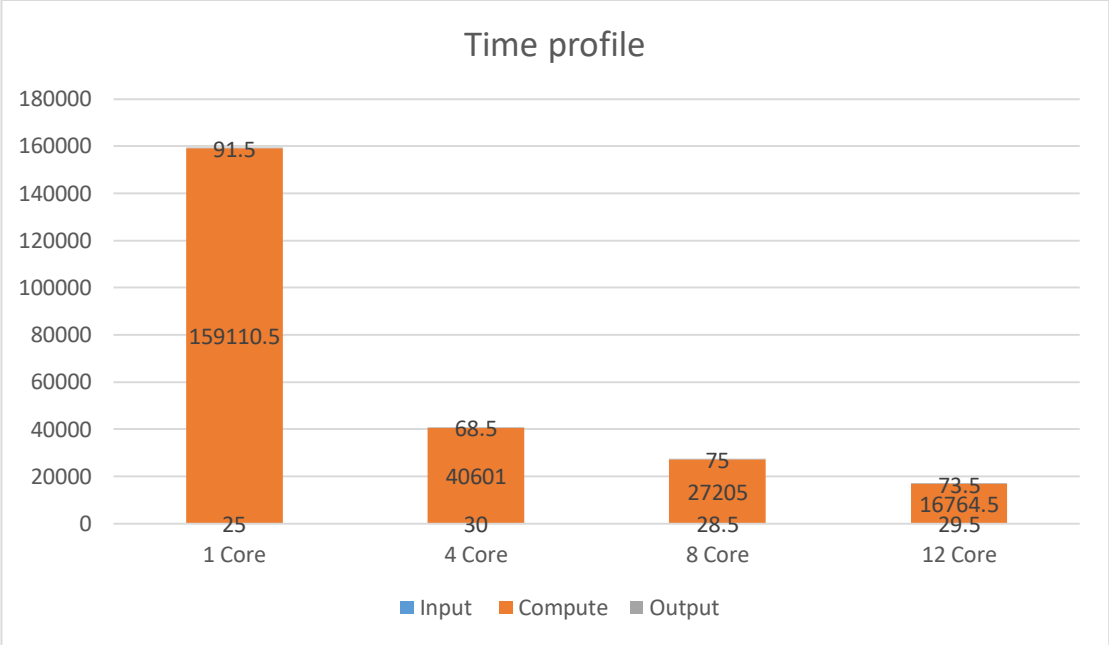



由圖中可以看出，實驗中的 Scalability 不錯，基本上可以呈穩定的倍數成長(可以從圖中的線性，也就是趨勢線，看出)，並且沒有很明顯坡度變緩的趨勢，雖然沒有到 12core 就會是 12 倍的程度。

就上面的 Time Complexity 來看 O ($\frac{V^3}{P}$)，應該要和 1/P 成正比，但是因為在 Computation 的時候中間會有 barrier，造成最外成的迴圈無法完全平行，

此外整體程式運行不只包括 computaion time 也有其他 IO 的時間，也就造成上圖無法完全成倍數成長的原因。

(c) Time profile：(左邊單位為 ms)




這張圖也是經過幾次運算後取平均的結果，可以看出隨著 core 增加計算時間大幅下降，這也合乎預期。整體來看 IO 不佔多少時間，仔細觀察會發現 input 會有稍微下降的趨勢，因為我是透過 threads 去讀 input，至於 output 沒什麼增減，都維持在某個數值左右，因為對 output 沒有做平行化的處理。

**Experiences/Conclusion：**

這次作業讓我對 Pthread 更加熟悉，一開始有嘗試使用 Omp 但是發現速度並沒有到非常快，我認為可能是因為在使用 Omp 的時候會切得太細而導致 overhead 增加，由於不太清楚他的底層運作方式，所以還是比較偏好使用 Pthread。此外這次作業我選擇使用 Floyd-Warshall 演算法，對於演算法加速的部分就比較沒有想法，雖然可以結合 Dijkstra's algorithm，在 edge 很少的時候會比較快，但是實作起來有點困難，而且不確定會加速多少，就沒有朝這個方向研究。所以這次優化主要是對 IO 著手，使用 mmap 的確可以加快一些時間，但是在 write 的時候也想使用 mmap 但是會遇到 error 所以並沒有對所有

IO 都做 mmap，在這部分覺得有點可惜。最後在 scoreboard 上有看到一個人的速度是大家的 2 倍快，因此蠻好奇他使用的方法。