OS HW2 Report **H34096102 謝欣蓉**

1. Dispatch works to the worker threads by row.

```
int k = id;
while(k < m1_row)
{
    for (i = 0; i < m2_col; i++) {
        int j;
        for (j = 0; j < m2_row; j++) {
            sum += (*thread_data).a[k][j] * (*thread_data).b[j][i];
        }

        (*thread_data).result[k][i] = sum;
        sum = 0;
    }
    k += NUM_THREADS;
}
```

當多個執行緒同時並運行時我用 while 迴圈讓每個執行緒能先由自己的 id 值作為開始矩陣乘法的行值，在做完該行的乘法之後會每次增加總執行緒數來做為下一個要乘法的行值。(ex: 執行緒數=4，那麼 id=2 的執行緒就會從第 2 行開始做乘法，乘完後下次由第 6 行開始，也就是 2+4)

2. Summary of the four charts

我覺得其實在做完四個測資之後有些值可能因為執行緒數量別不大所以分不太出來 elapsed time 的差別而且有時候在重跑之後兩執行緒跑出來的時間結果快慢會交換。不過我還是有觀察出一些每次都會發生的差別:

(1) number of threads is equal to the number of cores

→ 執行緒數=4 時永遠都會有最小的 elapsed time

→ 我覺得會出現這種情況可能是因為每個核心都能剛好處理一個執行緒這樣可以分散運行時間之外也可以少了交換執行緒到不同核心的時間。

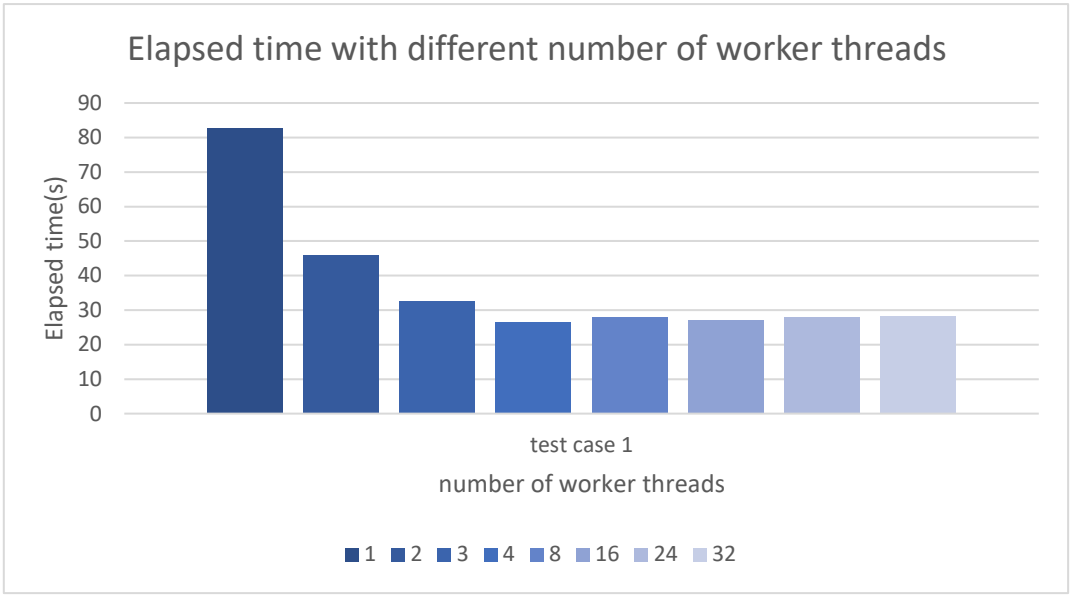
(2) number of threads is less than/greater than the number of cores

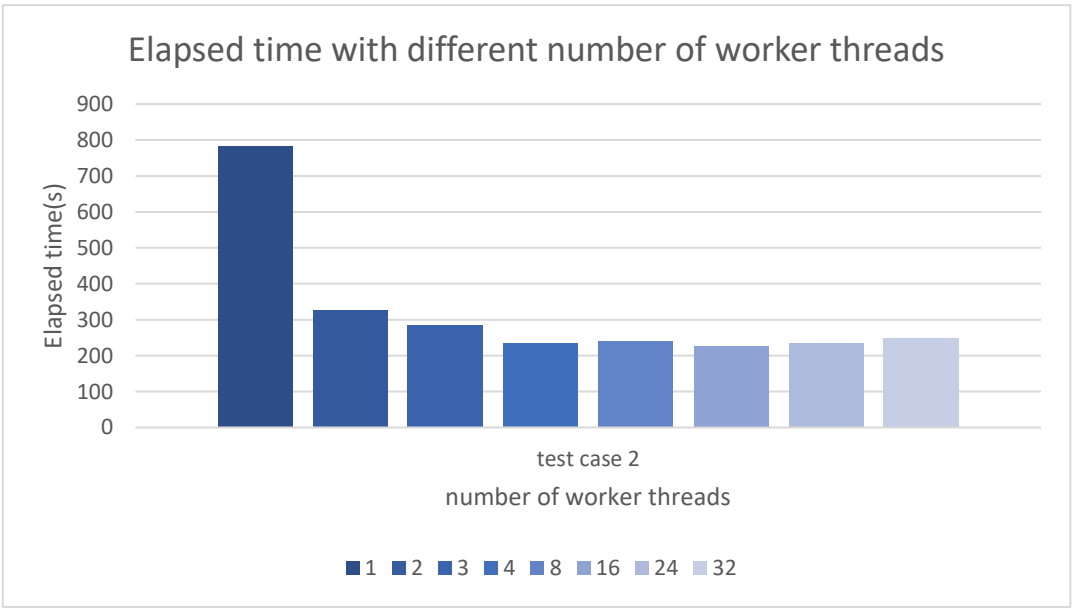
→ 共通點: 數量越少的執行緒會有越大的 elapsed time

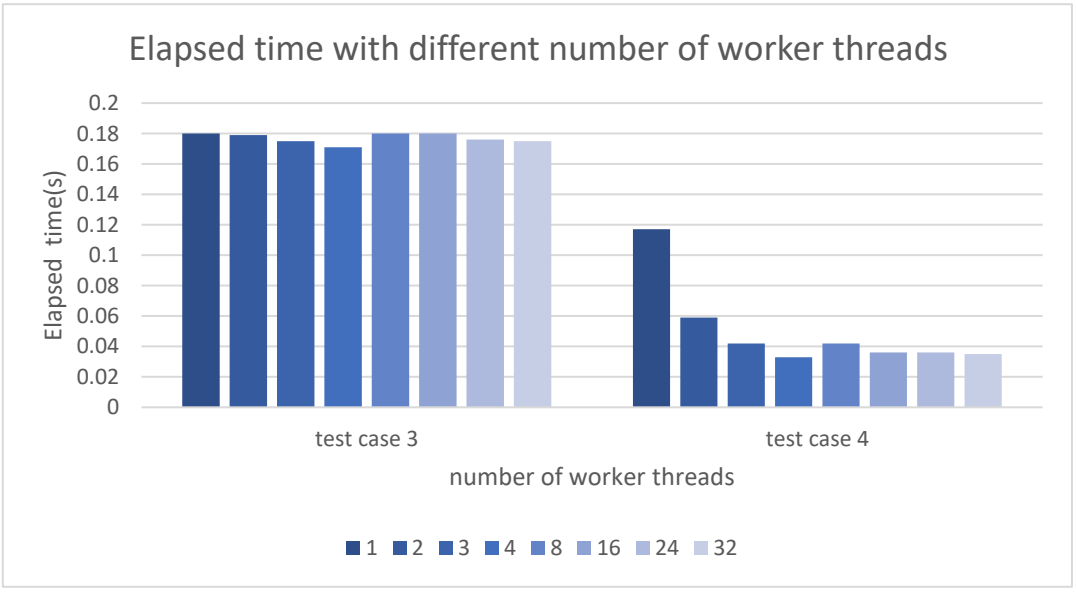
→ 差別: 執行緒數比核心數少時會隨著執行數量變多明顯看出 elapsed time 驟減, 而當執行緒數比核心數多時 elapsed time 的下降幅度不明顯

→ 從這樣觀察中發現當執行緒數比核心數多時, 每次測量的執行緒數量也差很多(每次都差 8 個)但反而 elapsed time 的下降幅度不明顯; 當執行緒數比核心數少時, 每次測量的執行緒數量僅差 1 卻能出現很明顯的 elapsed time 下降幅度。所以我覺得當有多出來的核心時, 執行緒的數量會大力影響 elapsed time, 就像是有多少執行緒就有多少核心在跑的感覺, 核心數越多就跑越快。但當執行緒數已經超過了核心數, 每個核心就要負責 2~8 個執行緒, 在運作的核心數相同時時間差影響就不大了。

(測試結果圖表在下一頁)


Elapsed time with different number of worker threads
Elapsed time(s)
90
80
70
60
50
40
30
20
10
0
test case 1
number of worker threads
1 2 3 4 8 16 24 32


Elapsed time with different number of worker threads
Elapsed time(s)
900
800
700
600
500
400
300
200
100
0
test case 2
number of worker threads
1 2 3 4 8 16 24 32


Elapsed time with different number of worker threads
Elapsed time(s)
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
test case 3
test case 4
number of worker threads
1 2 3 4 8 16 24 32