Report by E94091063 鄭鈞智

一、multithread matrix multiplication 實作方法：

我是將 m1 的 row 數平分給各個 thread 來做矩陣相乘，設 m1_i 為 m1 的總列數，N 為總 thread 數。

方法一：

第 1 條 thread 處理第 1~m1_i/N 列的矩陣相乘，第 i 條 thread 處理第 (m1_i/N)*i+1~(m1_i/N)*(i+1)列的矩陣相乘，第 N 條 thread 則處理第 (m1_i/N)*(N-1)+1~m1_i 列的矩陣相乘，即如果 m1_i 無法被 N 整除時，多餘的列數都會直接丟給第 N 條 thread 處理。所以如果一個 2048*2048 的矩陣要切割給 24 條 thread 時，第 1~23 條 thread 會處理 2048/24=85 列，第 24 條 thread 會處理 2048/24+2048%24=85+8=93 列

方法二：

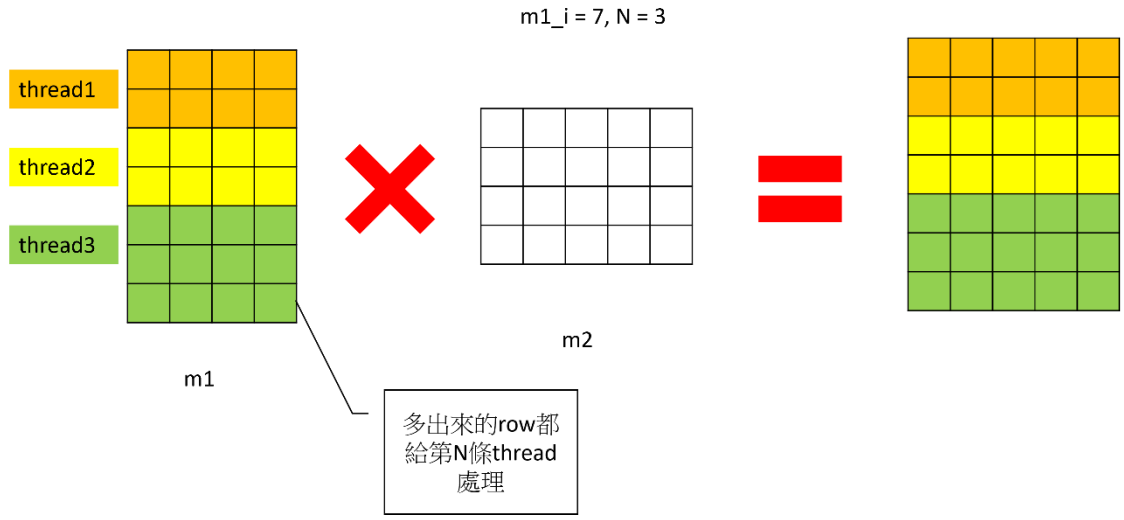
若 m1 的 row 數小於總 thread 數，則會將 m2 的 column 數平分給各 thread 做矩陣相乘。

方法三（同方法一）：

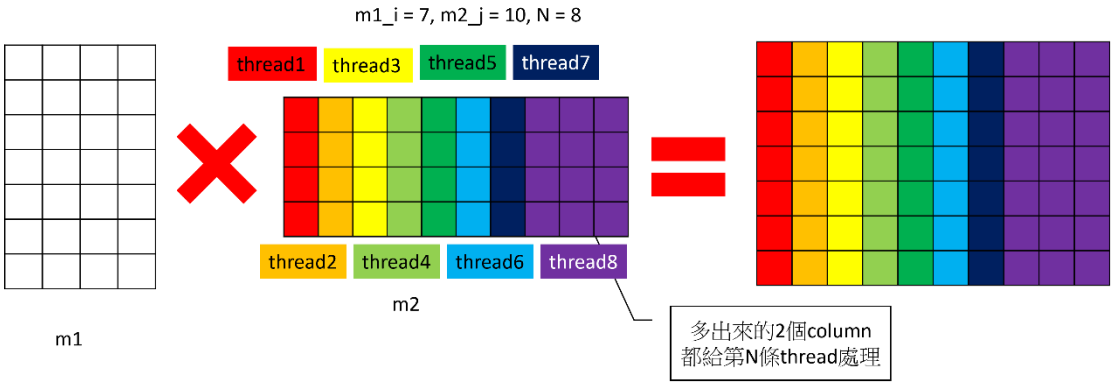
若 m1, m2 的 row, column 數皆小於 thread 數，則照方法一直接實作

示意圖：

方法一

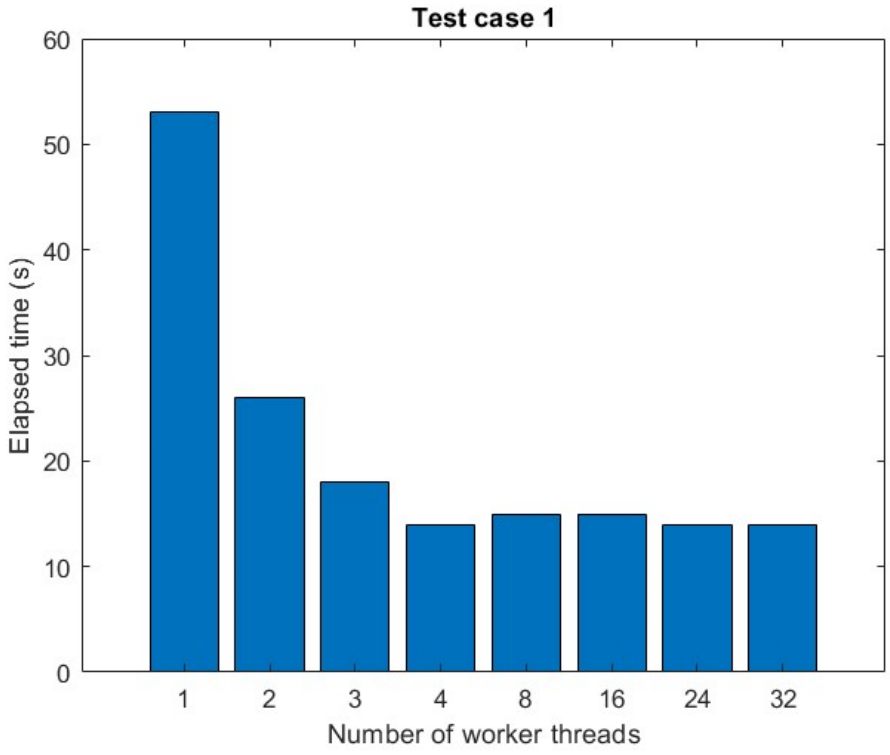


方法二

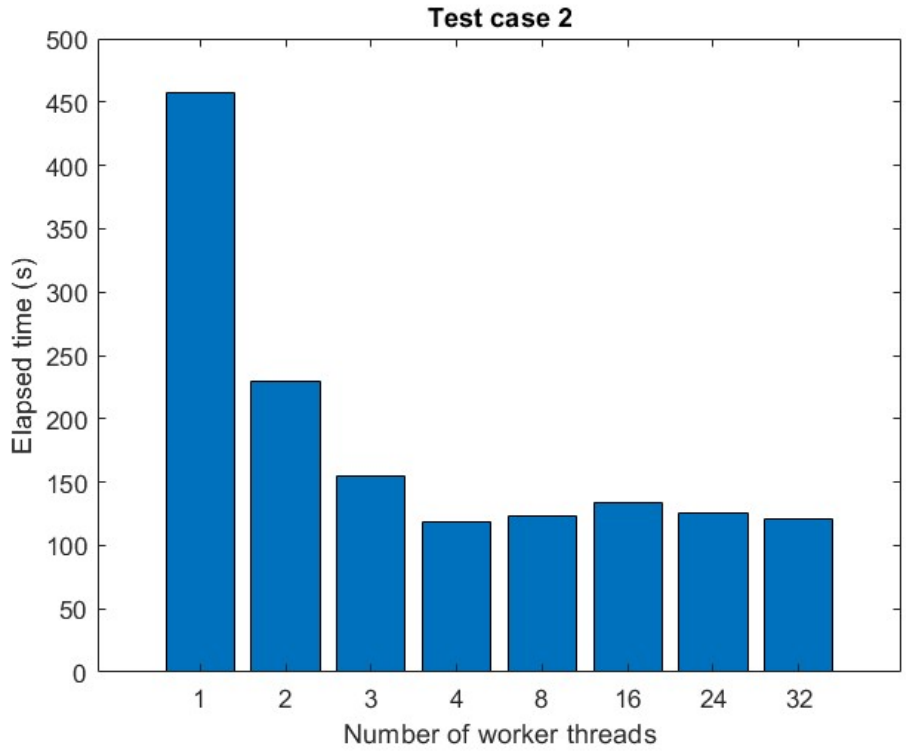


二、各 case 之 execution time v.s. the following worker thread numbers:

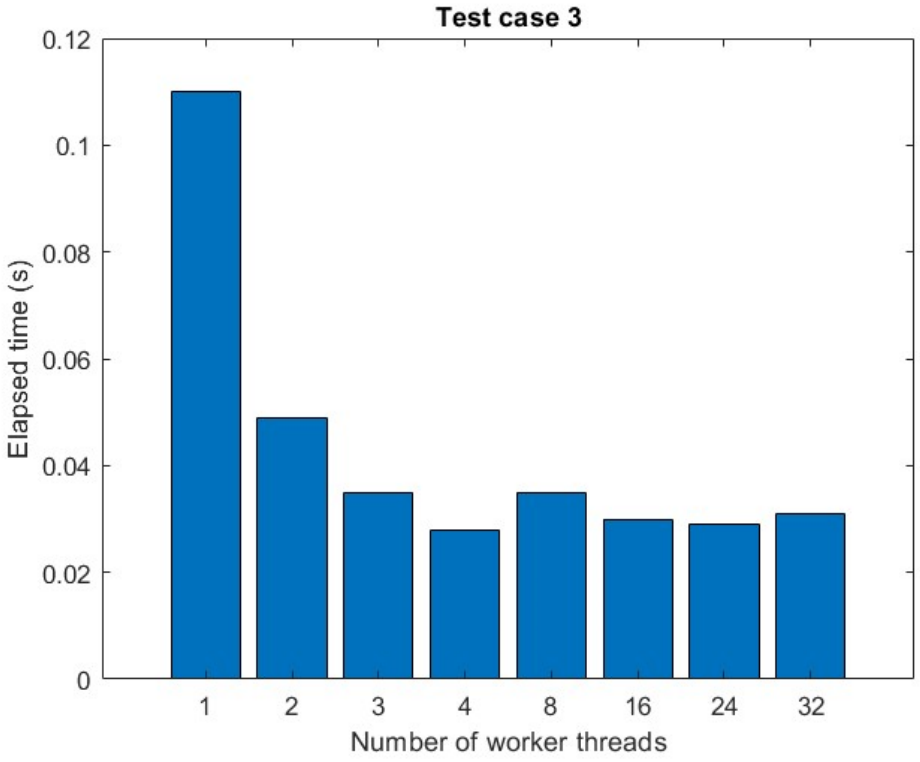
Case1:



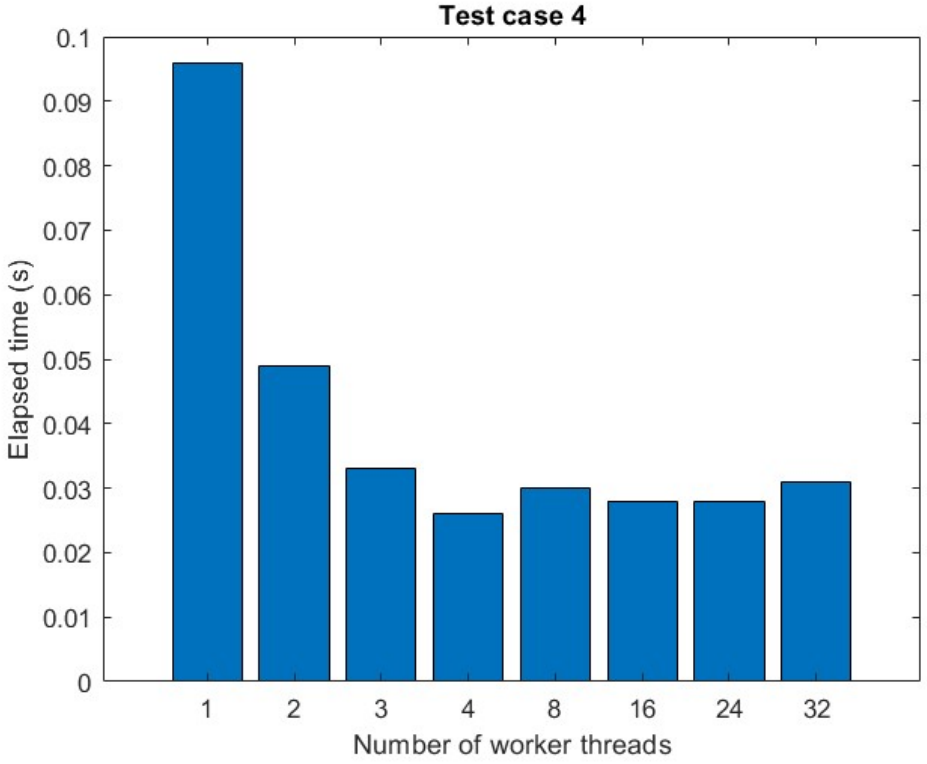
Case2:



Case3:


Test case 3
Elapsed time (s)
0.12
0.1
0.08
0.06
0.04
0.02
0
1
2
3
4
8
16
24
32
Number of worker threads

Case4:


Test case 4
Elapsed time (s)
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
1
2
3
4
8
16
24
32
Number of worker threads

三、Summarize the four charts.

1. 本次使用 4 cores 做為虛擬環境。由 4 個 case 可知，當 thread 數小於 core 數時，execution time 會變得非常長，尤其 case2 的兩個 4096*4096 矩陣相乘真的讓我等非常久。主要原因是因為多個 core 可以同時運行不同的 thread，所以當 thread 數為 1 時，可以說相當於有 3 個 core 是空閒著的。以 case2 為例，因為執行的時間非常久，所以可以明顯看到當 thread 數為 2 時，elapsed time 幾乎是 thread 數為 1 的一半，而 thread 數為 4 時，elapsed time 為 thread 數為 2 的一半。
2. 當 thread 數大於 core 數時，elapsed time 幾乎與 thread 數=core 數時持平，甚至會更久。這是因為執行環境為 4 個 core，但跑多於 4 條 thread 時，就無法讓一個 core 專心處理一條 thread，也就是說 thread 再怎麼多，還是只有 4 個 core 在平行處理，所以無法增加運行速度。