作業系統

HW2 Report

資訊系 113 郭立晨 F94081076

分配工作的方法:

讀取完兩個矩陣後，舉例: axb, bxc，答案的維度會是 axc，我會依照 a、c 誰比較大來決定要沿著列還是行切分。假設是 a 比較長，則會沿著列切分工作，反之若 c 大於等於 a 則是沿著行切分工作。如此才能平均的分散工作給所有 threads，面對比較極端的情況，舉例，2048*1 、1*2048，才不會所有工作集中在一個 thread 身上。若是無法完美分配，有餘數的情況，我安排最後一個 thread 把剩下的工作全部做完。

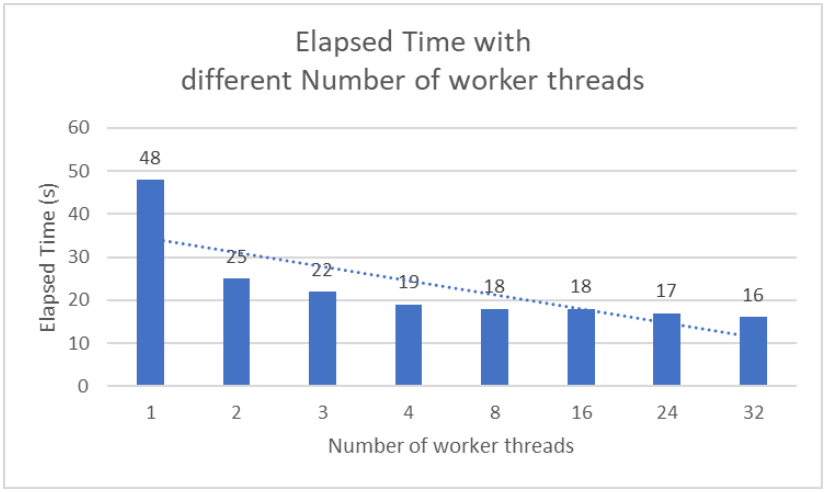
測資對應 Threads 數量的表格:

	1	2	3	4	8	16	24	32
Test1	48	25	22	19	18	18	17	16
Test2	875	627	452	422	476	467	493	492
Test3	1	1	0	0	0	0	0	0
Test4	0	0	0	0	0	0	0	0

時間單位: 秒(sec)

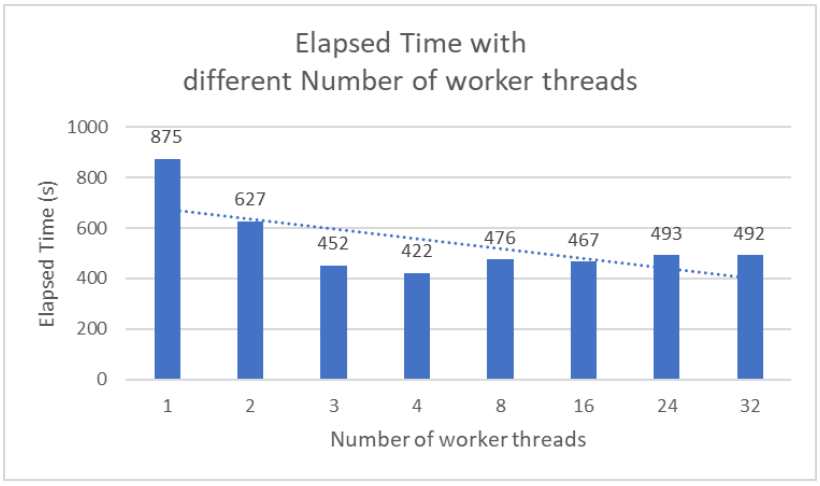
圖表:

- Test case 1



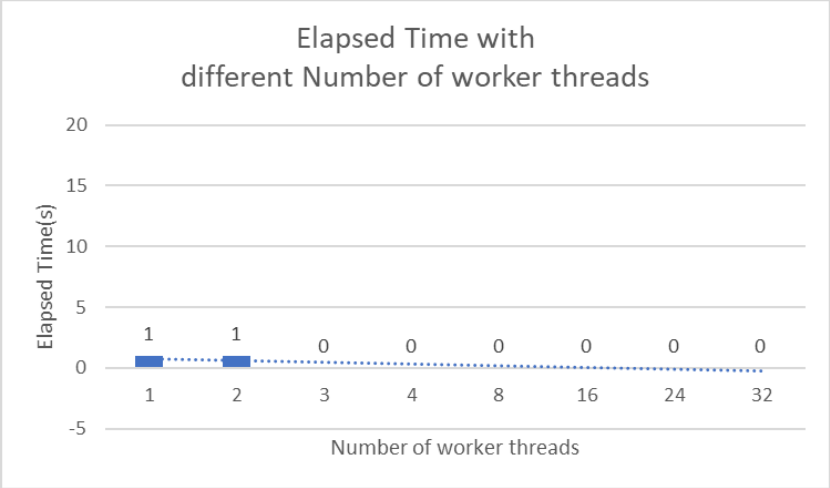
可以看出 thread 數量越多花費時間越少，一直到與核心數量一樣之後便趨於平緩，變化不大。(虛線為趨勢線)

- Test case 2

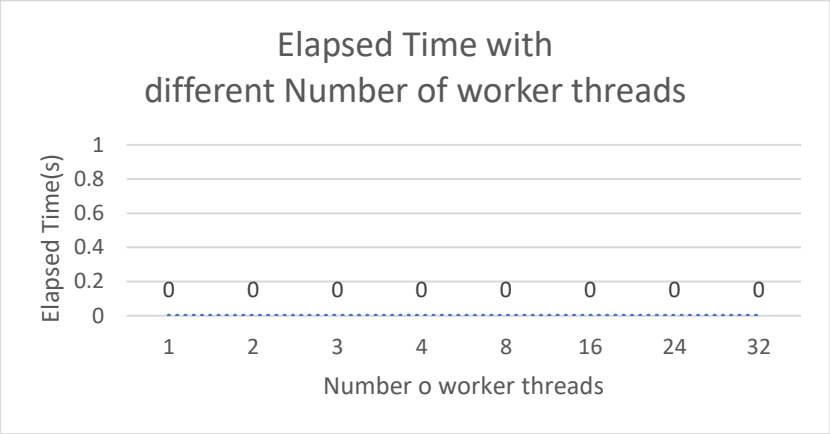


從此筆測資中最能看出 context switch 造成的時間成本，隨著 threads 數量超過核心數量越多有上升的趨勢。(虛線為趨勢線)

- Test case 3



- Test case 4



測資三四，因為結果差不多，故統一做說明，由於精度規定在秒數(sec)，執行的時間大部分為零秒，除了測資三在使用一、兩個 threads 的時候有測到一秒的情況。因此還是可以些微的注意到在測資一二發現的現象存在，只是可能要透過更精確的時間測量才能看出差異。

問題:

1. **What happen if the number of threads is less than the number of cores. Why ?**

   由於測資三、測資四在時間精度規定為秒的情況下執行時間大部分為 0，故難以從圖形中看出明顯的趨勢。但是從測資一、二中可以發現使用的 threads 數量越接近核心的數量時，速度是越來越快的，大致的趨勢可以從圖中的虛線得知。因此，threads 數量低於核心數量，執行時間是比較長的。因為當 threads 的數量比較少時，核心不會被矩陣運算的工作占滿，因此無法最大化整體的效率。

2. **What happen if the number of threads is greater than the number of cores. Why ?**

   若 threads 數量超過核心的數量，從測資一、二圖中可以觀察到，效率不會有明顯的提升，甚至會有變差一點的狀況。因為此時需要 CPU 的排程來決定現在要執行哪一個 thread，而執行的時間就會因為 context switch 以及更新暫存器資料等原因而浮動。若是進行 context switch 所花費的時間更多於執行該 thread 的時間，那效率就會變得更差，資源浪費過多。

3. **Anything else you observe**

   從各個測資產生的圖形可以觀察到，要執行的工作越繁重，越能看出多個 threads 之間由 context switch 造成的 CPU 資源浪費，因此若是可以為不同的任務作不同 threads 數量的安排，可以最大化核心使用的效率。

   另外一個發現則是，測資一、二的計算時間差非常多，在經過與同學討論後，認為有可能是 Row major 的記憶體對於要一次取出一個 column 的資料是相對花費時間的，因此將矩陣做轉置後以每次取出 row 的資料做運算來取代原本的方法，應該會讓運算時間降低。