SP_hw4

資工二 B03902042 宋子維

n	n/100	n/25	n/10	n/5	n/2	n
100	(1) 0.007 0.000 0.011	(4) 0.003 0.000 0.004	(10) 0.001 0.000 0.002	(20) 0.001 0.000 0.001	(50) 0.001 0.000 0.001	(100) 0.001 0.000 0.001
10000	(100) 0.024 0.026 0.004	(400) 0.019 0.017 0.004	(1000) 0.013 0.014 0.003	(2000) 0.010 0.007 0.004	(5000) 0.008 0.008 0.000	(10000) 0.006 0.006 0.000
1000000	(10000) 1.372 1.311 0.072	(40000) 1.090 1.060 0.032	(100000) 0.933 0.906 0.028	(200000) 0.810 0.791 0.020	(500000) 0.617 0.606 0.012	(1000000) 0.469 0.458 0.012
10000000	(100000) 18.522 13.060 0.642	(400000) 12.160 10.551 0.546	(1000000) 11.827 9.168 0.429	(2000000) 9.773 7.787 0.356	(5000000) 7.423 5.940 0.200	(10000000) 5.481 4.704 0.147

(The segment_size is in the clauses of each entry, and the remaining line represents real, user and system time respectively.)

從表格中，可以看出當segment_size越大，也就是segment的數量越少時，程式不管在real time或者user time都較少，而且real time大部份比user time大，可是一般而言，launch越多的thread，應該能加快程式。我認為，由於這次的作業要求要在thread的start rountine裡面輸出處理的過程，為了不讓輸出亂掉，所以必須用mutex包住I/O的部份，而I/O又比較耗時，因此，thread會佔用CPU，直到I/O結束才能unlock mutex，讓其它thread使用CPU。其次，每當處理完一個round，在main thread裡我都用了for loop join其它thread，故當segment數量越多，thread數量也越多，導致main thread block在join，使得real及user time增加。

於是，我將thread裡的mutex以及I/O關掉，並跑了相同的測資，得到了以下結果

n	n/100	n/25	n/10	n/5	n/2	n
10000000	(100000) 1.355 2.652 0.175	(400000) 1.200 2.682 0.094	(1000000) 1.216 2.761 0.044	(2000000) 1.390 2.631 0.032	(5000000) 1.576 2.268 0.040	(10000000) 2.258 2.243 0.016

從中可以發現，除了程式比I/O關掉前快很多之外，real time和user time也大致上隨著segment的數量減少而增加，而且real time幾乎都比user time小，這樣子的結果比較能代表multithread對程式的加速。同時，觀察兩者real time和user time的大小，先看看Description：

Real is wall clock time - time from start to finish of the call.

User is the amount of CPU time spent in user-mode code (outside the kernel) *within* the process.

前者因為單一個thread的mutex+I/O佔用過久的CPU，導致multithread平行處理的效果不佳，故real time > user time(total CPU time)；而後者，因為多CPU得到有效的分配，不同的thread能達到平行處理，因此real time < user time。