多處理機平行程式設計 作業一 report

F74109032 陳均哲

平行化與未平行化的差異

平行化的概念是把原先一個未平行化的問題切分成多個process讓他們同時執行減少時間。而在這次作業是利用p2p communication的方式讓多個process之間互相傳送資訊。因此當process數量增加，程式執行速度會變快，但相對也會在communicaiton上花較多時間。

Problem 1

1.時間比較

Serial Programming: 0.001438 sec
Parallel Programming:

n=	4	8	16	32	40
time(sec)	0.000578	0.000355	0.000403	0.000855	0.001797

> 未平行化的結果使用time.h裡面的clock()來計算
> 平行化的結果則使用以下指令測試執行，(hostfile分為16,8,8,8)
> 指令: `mpiexec -f hosts -n 32 ./a.out`

2.結果分析

從結果來看，當分成4以及8個process平行運算，執行時間有變快；但是當分成16~40個process時間運算時，執行時間則是漸漸變慢，而當40的時候則比原先未平行化的還慢。針對這個結論我有以下幾個推測：
(1)這個問題是個迴圈從0~65536執行checkCircuit判斷。而因為迴圈裡面次數少，本來的執行時間就不高，因此當把這個問題分段平行化並不會產生太大的變化。
(2)當分成越多個process執行，process之間的p2p communication所花時間就越長，以tree-structure的邏輯來看，communication的時間複雜度大致上是O(log(n))。

綜合以上兩點從表格數值分析，當n=16開始，communication所花的時間大於平行化減少的時間，因此執行速度會漸漸變慢，到n=40時比未平行化的結果還慢。

Problem 2

1.時間比較

(單位:sec)

tosses	Serial	4	8	16	32	40
10^6	0.028969	0.009086	0.021114	0.034808	0.033219	0.032656
10^7	0.301771	0.182401	0.103013	0.084812	0.047338	0.041336
10^8	2.944064	1.360270	0.875967	0.515003	0.369936	0.315275

tosses	Serial	4	8	16	32	40
10^9	30.694831	11.867415	6.364261	4.916112	3.481626	3.074178
10^10	272.706166	113.134916	64.656185	50.234759	37.801532	29.175620

> 當tosses次數為10^6以上，就可以相當穩定地得到較接近真實π的數值，因此測資設定 10^6~10^{10}

2.結果分析

這個問題使用的測資範圍相較於第一題的65536大很多，因此平行化所帶來的效益會比較明顯。

(1)10^6時，因為迴圈次數還不夠多，因此只有process為4的情況比較快，8開始減慢但還是比未平行化的快，而分成16以上時則比未平行化還慢。當迴圈次數不多的情況下做平行化，communication所花的時間容易大於平行化減少的時間，因此產生的結果並沒什麼參考價值，也沒有做平行化的必要。

(2)10^7以上時，產生的結果漸漸開始隨著process數量增加執行時間減少。10^7的情況下這個現象還不明顯也不穩定，但當迴圈次數來到10^9 ~ 10^{10}時，執行時間明顯隨著process數量增加而減少。

(3)還有一個現象是在不考慮process間communication的情況下，toss次數每變成10倍，執行時間也會大致上也會變成10倍。而表格上10^8 ~ 10^{10}的情況下比較明顯，因為communication時間遠小於迴圈的計算時間。10^6 ~ 10^7則是communication時間相較於迴圈計算時間顯得太多。

總結

從這兩題測試下來的結果可以看出平行化的問題大多是在不同process數量間取程式計算量與communication所花時間的平衡。當計算量不大的情況，process設比較小或是甚至不用平行化效果就很好。而當計算量較大的情況下，使用平行化則是一個可以明顯降低程式執行時間的好方法。