System Programming
Programing Assignment 4 Report

電機三 金延儒 B02901107

January 18, 2016

1 測試

測試講求的是準確，為了使測出來的時間更準確，我大概有以下幾個方面的改進。

首先是輸出的部分，一開始是把本來程式的 output 重導到/dev/null，但後來發現，即使導到/dev/null，依舊會多耗掉一些時間在 user time。讓我們看看測資 $n = 5 \times 10^7$ 時兩種的差別。其中箭頭代表時間隨著 thread 數的增加而增加（↗）、減少（↘）或持平（→）。

(sec)	# of threads	1	2	5	10	25	100
Delete output	real ↘	20.212	13.734	11.912	11.786	11.672	11.734
	user →	20.070	20.192	20.338	20.240	20.068	19.982
	sys ↗	0.150	0.130	0.162	0.188	0.220	0.366
To /dev/null	real ↗	27.230	25.018	28.216	31.180	35.326	42.784
	user ↗	27.078	30.944	37.132	40.962	45.812	53.208
	sys ↗	0.168	0.192	0.254	0.282	0.440	0.840

拿 redirect to /dev/null 跟 delete output 相比，我們可以觀察到 system time 雖然有增加，但是非常不顯著，最主要是 user time 部分的增加，又如果這部分增加的 user time 是可以平行化的，那麼均攤下來可能不會差非常多。但不幸的是，我們可以看到 terminal 輸出無法平行化，所以這會使得 real time 急遽增加，甚至原本因為 threading 帶來的效益也沒了。也就是說：在要輸出過程的情況中，threading 在時間上的花費反而會更差，主因在於其中輸出的難以平行化。

再來是測資產生，餵給主程式的部分，如果我們將主程式的測資輸出到一個檔案裡面，測試時再由主程式直接去讀取那個檔案，這樣會造成一個問題：檔案在讀寫是要 trap into system，會多花一些 system time，並且大幅減少效率。這個的解決方法很簡單，就是把產生測資的程式直接 pipe 給 merger，這樣子直接在記憶體之間傳送可以大幅降低 IO 的時間。

再接下來，若我們用系統的 time 指令來計算時間，我們只能知道包括亂數產生以及排序的 real, user, system time，並無法知道個別的資訊。為了要能夠準確亂數產生以及排序個別的資訊，最好的方法是另外自己寫一支測試，然後在每筆測資就 fork 出一個 process 來跑主程式，而測試程式除了可以用來更準確的偵測 child 所花的時間，更可以控制流程、紀錄統計資料，是一個比較適當的方法。

另外，為了使測出來的時間更具有可信度，在同一個資料量，同一個 thread 數下，我也用不一樣的測資去跑，最後加總平均。這同樣也是可以用自己另外寫的程式完成的。

2 結果

以上是我在測試程式上的改進，而對結果造成影響的討論，接下來討論不同測資大小及 thread 數目下，平行化的 merge sort 執行的時間結果。(這是把 ouput 去掉，然後每筆跑五次的加總平均)

(sec)	# of threads		1	2	5	10	25	100
Real time	$n = 10^2$	$\rightarrow$	0.000	0.002	0.000	0.000	0.004	0.004
	$n = 10^4$	$\rightarrow$	0.004	0.002	0.004	0.006	0.002	0.008
	$n = 10^6$	$\searrow$	0.346	0.244	0.226	0.222	0.224	0.220
	$n = 10^7$	$\searrow$	3.800	2.634	2.310	2.286	2.264	2.278
	$n = 5 \times 10^7$	$\searrow$	20.212	13.734	11.912	11.786	11.672	11.734
User time	$n = 10^2$	$\rightarrow$	0.000	0.000	0.000	0.000	0.000	0.000
	$n = 10^4$	$\rightarrow$	0.004	0.004	0.002	0.000	0.006	0.002
	$n = 10^6$	$\rightarrow$	0.340	0.344	0.344	0.336	0.338	0.326
	$n = 10^7$	$\rightarrow$	3.782	3.790	3.808	3.784	3.748	3.786
	$n = 5 \times 10^7$	$\rightarrow$	20.070	20.192	20.338	20.240	20.068	19.882
Sys time	$n = 10^2$	$\rightarrow$	0.000	0.000	0.000	0.000	0.000	0.000
	$n = 10^4$	$\rightarrow$	0.002	0.000	0.002	0.002	0.002	0.002
	$n = 10^6$	$\rightarrow$	0.004	0.002	0.002	0.012	0.008	0.008
	$n = 10^7$	$\nearrow$	0.022	0.036	0.034	0.052	0.060	0.072
	$n = 5 \times 10^7$	$\nearrow$	0.150	0.130	0.162	0.188	0.220	0.366

首先我們可以觀察到，在 thread 為 1 時，也就是完全沒有平行話時，總時間

大概就是 user time + system time。而在 user time 以及 system time 之中，很明顯地，user time 佔了顯著的地位。也因此，即使在 n 比較大，system time 隨著 thread 數而增加時，real time 依舊不會有明顯的增加，反而會因為 user time 分配在不同處理器下而有更好的表現。

此外，我們可以看到雖然整體而言，實際執行時間隨著 thread 的增加而減少，但是當 thread 數太多，例如 100 時，執行時間反而沒有 25 來得好，我想，最主要的原因就是 system time 的增加。雖然效率的降低並不是十分顯著，但是這依舊是可以再仔細考慮，並且 trade-off 的關鍵。