[作業系統概論 HW11]

409410025 邱 X 恩

---

[1] 解釋：sched_latency_ns 和 sched_min_granularity_ns

sched_latency_ns : 表示一個 running queue 裡面所有的 process 運行一次的週期。如果 process 比較多，context-switch 也會比較多，每個 process 分配到的時間就會比較少。

sched_min_granularity_ns : 表示一個 process 執行一次所需要的最少時間。若是將這個值設得較小，process 執行一次就不用那麼多時間，context-switch 的次數就會比較多。

---

[2] 設計實驗，說明 context-switch 的次數與效能的關係

(a) 系統環境

硬體環境

Memory	3.8 GiB
Processor	Intel® Core™ i5-10300H CPU @ 2.50GHz × 2
Graphics	SVGA3D; build: RELEASE; LLVM;
Disk Capacity	53.7 GB

軟體環境

OS Name	Ubuntu 20.04.2 LTS
OS Type	64-bit
GNOME Version	3.36.8
Windowing System	X11
Virtualization	VMware
Software Updates	〉

Cpu 速度

```
CPU MHz:                              2496.000
```

[2] 設計實驗

(a)系統環境

Linux 初始設定 sched_latency_ns / sched_min_granularity_ns

```
root@vm:/sys/kernel/debug/sched# cat latency_ns
12000000
root@vm:/sys/kernel/debug/sched# cat min_granularity_ns
1500000
```

(b) 開始實驗

執行方式 ：

./reportChildStat ./cpu

Linux 設定 ： **將 sched_latency_ns / sched_min_granularity_ns 都調低為 100000**

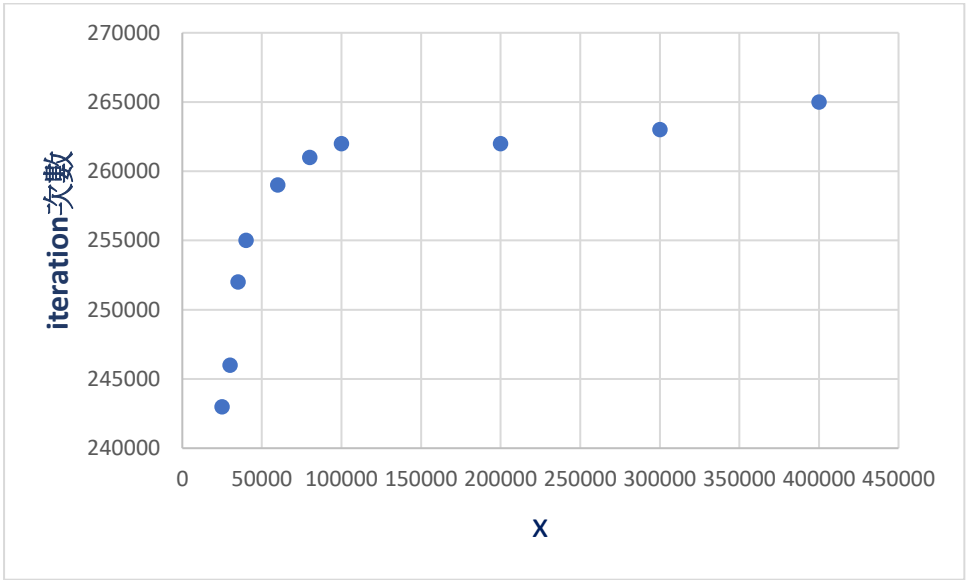
當 **iteration % X=0** 時就會執行 sleep()，進而產生**自願性**的 ctx-sw，

透過更改 x 的值來觀察 context-switch 跟 iteration 之間的關係。

實驗數據

x	25000	30000	35000	40000	60000	80000	100000	200000	300000	400000
iteration次數	243000	246000	252000	255000	259000	261000	262000	262000	263000	265000
自願ctx-sw	163816	147390	114650	98285	65531	49148	32768	16383	4752	1
非自願ctx-sw	542	885	486	612	474	528	594	510	416	597
ctx-sw	164358	148275	115136	98897	66005	49676	33362	16893	5168	598

實驗 x-y 散佈圖



[2] 設計實驗

(b) 開始實驗

實驗結果說明

X 軸 : iteration % x 的 x (**操縱變因**) || Y 軸 : iteration 次數

當 iteration%x 中的 x 越大，可以在散佈圖看出來，iteration 的次數大致上會跟著變大。Iteration 的次數表示程式執行幾回合，也就是和程式的效能有關係。當 x 越大，此程式就越不容易 sleep，也就是自願性的 ctx-sw 次數下降。當 ctx-sw 的次數下降，iteration 的次數增高，這就表明了 ctx-sw 的次數越低，會有比較好的效能。

原因在於 ctx-sw 會有更新 Cache/TLB 的 OVERHEAD 有關。

如何設計實驗才可以得到更多的 ctx-sw 與效能的數據 ?

從上面的實驗可以看出 ctx-sw 的次數和效能的關係，因此我這邊會利用調整 sched_latency_ns / sched_min_granularity_ns 的方式，去增加 ctx-sw 整體的數量，如此一來更可以看得出來 ctx-sw 之間的關係。

為何選擇上述的執行方式 ?

./reportChildStat ./cpu

將 sched_latency_ns / sched_min_granularity_ns 都調低為 100000

當每個 PROCESS 分配到的時間越少，越容易 CTX-SW，

也會比較容易觀察整體數據的變化。

當 **iteration % X=0** 時就會執行 sleep()，進而產生自願性的 ctx-sw，

透過更改 x 的值來觀察 context-switch 跟 iteration 之間的關係。

(c)

預估 context-switch 的數量到達多少的時候，

程式碼的執行時間會超過一個小時 ?

根據上面的實驗，經過我在 vmware 的測試，

每 10000 次 ctx-sw，

總共約需要 8 秒左右的時間。

也就是 1 秒大約有 1250 次 ctx-sw。

當執行時間超過 3600 秒，

我預估 ctx-sw 的數量大約為 **4500000** 次以上。