2020_OS_Fall_HW2: ETL process

學號 : F14076083
姓名 : 魏湧致
系級 : 資訊 111

開發環境 :

- 使用 Virtual Machine
- OS : Ubuntu 18.04.2 LTS
- CPU : Intel® Core™ i7-7700HQ CPU @ 2.80GHz × 3
- Memory : 3.9 GiB
- Programming Language(version) : C++ 11

程式執行時間 :

- 使用 time 指令得到的執行時間
  虛擬機環境為 3 個核心，無法測試設定 7 個 thread 以上的 output 正確性

Single Thread :		Two Threads :	
real	10m21.735s	real	5m2.136s
user	4m21.984s	user	2m34.247s
sys	5m53.613s	sys	3m5.711s

Three Threads :		Four Threads :	
real	3m21.962s	real	3m3.300s
user	2m32.072s	user	2m32.002s
sys	3m34.720s	sys	3m40.998s

Five Threads :		Six Threads :	
real	2m23.521s	real	2m14.058s
user	2m18.875s	user	2m16.833s
sys	3m45.063s	sys	3m50.984s

程式開發與使用說明：

- **你是如何開發這支程式，程式在處理資料的流程及邏輯為何 :**
  程式分為兩個 stage(function)，function read_csv 用來讀取 input.txt，使用 getline 將 input.txt 的一行放入 string，再依據”|”來切割，將讀取到的數字依序放入 linked list 中，function write_json 用來將資料依照 json 的格式寫入檔案，每個 read function 會產生一個 linked list，會由 write 一一對應來

寫入檔案中。
而依據執行時輸入的參數，若是偶數的話兩個 function 就各分配一半的 thread，基數的話 read_csv 則比 write_json 多分配一個，因為 read 和 write 會一一對應，故若 thread 數量為基數時要先讓 read 執行，等 read 的其中一個 thread join 後再 creat 來寫最後一個 linked list。

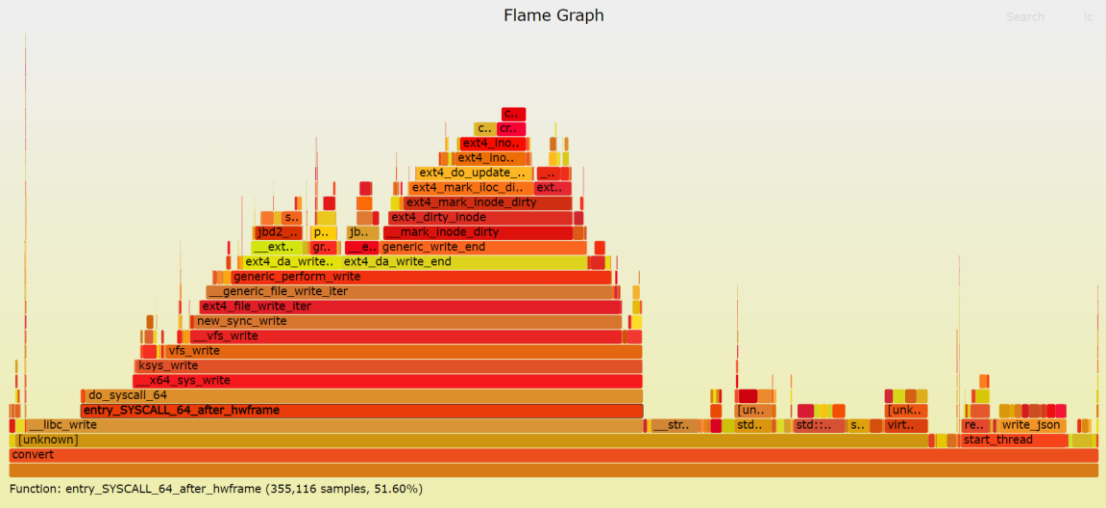
- **你的程式該如何使用，請詳細說明執行的步驟：**
  Compile : g++ -pthread -o convert convert.cpp
  Run : ./convert [threads]

效能分析報告：

- **請觀察程式執行期間各個 Stage 對電腦資源使用情形：**
  Single Thread 時 top 指令與產生的火焰圖如下

```
top - 17:58:06 up  3:49,  1 user,  load average: 0.98, 0.74, 0.62
Tasks: 242 total,   2 running, 196 sleeping,   0 stopped,   0 zombie
%Cpu(s): 19.3 us, 32.9 sy,  0.0 ni, 46.9 id,  0.7 wa,  0.0 hi,  0.2 si,  0.0 st
KiB Mem :  4030676 total,   118032 free,  3677572 used,   235072 buff/cache
KiB Swap:  1942896 total,  1352184 free,   590712 used.    93912 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
15320 pd2       20   0 3080848 2.925g   2996 R  99.7 76.1   0:51.44 convert_3
 2461 pd2       20   0 3549464 181608  56224 S   2.3  4.5   5:12.89 gnome-shell
```

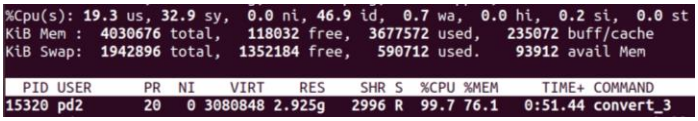


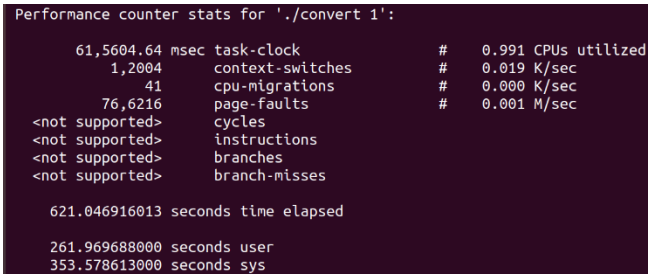
從 top 指令可以看出 read stage 主要運用到的資源是讀檔時會用到 disk 與 memory 及分割字串時會用到 CPU。另外由火焰圖也可以觀察到程式大部分的時間都花在 Function : __x64_sys_write，也就是 write_json 這個 stage，這個 stage 運用到的資源將資料寫到 Disk。

- **請觀察並比較不同執行緒（thread）數量下，程式的執行狀況、系統資源的使用：**
  因為是用 linked list 存放讀入的資料，故 1GB 的 data 使用到的 memory 約

為 3GB，而因為讀入的資料相同，所以不管用幾個 thread 跑都是一樣的使用量，設定不同 thread 只會影響 CPU 使用率。

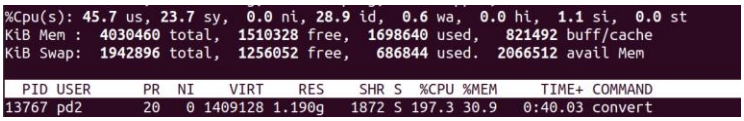
Single Thread:

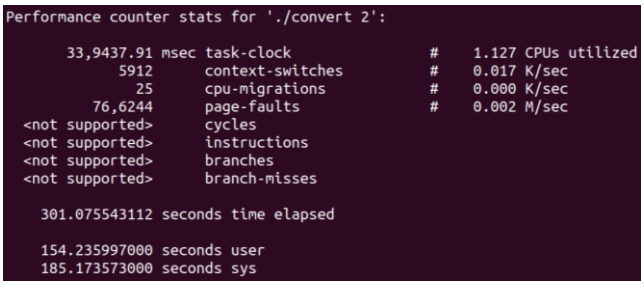




只使用一個 thread 時執行時間約為 10 分鐘，整體 CPU 使用率為 20%，執行時用到了 0.991 個 CPU。

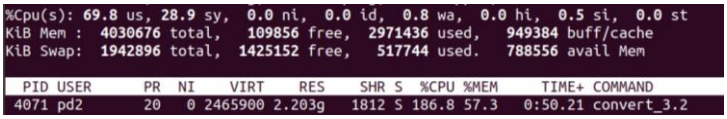
Two Threads :





使用兩個 thread 時執行時間降為約 5 分鐘，因為同時讀寫的關係，所以執行時間大幅降低，CPU 使用率上升到 45.7%，用到了 1.127 個 CPU，而 top 指令的%CPU 因為是各個 thread 累加，故來到了 200%。

Three Threads :



使用三個 thread 時執行時間為 3 分 20 秒，CPU 使用率約落在 70%，用到

了 1.820 個 CPU，也因為使用到較多 thread，所以 CPU 時間會比實際執行時間還要長，

- **請觀察系統效能以及 OS 是如何服務我們的程式，並做出結論:**
  因為這次的程式最主要是在讀檔以及寫檔，memory 部分因為是用 Linked List 的關係所以 1GB 的檔案使用了約 3GB 的記憶體，CPU 使用量會隨著使用 Thread 的數量提高而升高，多增加一個 Thread 約會增加 0.5 個 CPU 使用量，與一個核心內有兩個 Threads 相符合，而 OS 幫我們做了許多系統呼叫來執行大量的 I/O。

  結論：這次的作業是要寫一個 Multi-Thread 的程式來將 CSV 檔轉成 JSON 檔，自己決定要分配多少 Thread 給哪些 function 使用，Single Thread 時執行時間約為 10 分鐘，多了一個 Thread 後，讀檔案並處理資料與寫檔平行運作的情況下讓執行時間降為約 5 分鐘，使用多執行緒來平行執行程式雖然要花較多時間謹慎考慮各個 stage 間的相互關係，但程式的執行時間也會因此大幅降低。