HW1

電控碩一 309512074: 黃柏叡

hardware specification:

```
ray@ray-ubuntu:~$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   39 bits physical, 48 bits virtual
CPU(s):                          16
On-line CPU(s) list:             0-15
Thread(s) per core:              2
Core(s) per socket:              8
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           165
Model name:                      Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
Stepping:                        5
CPU MHz:                         2900.000
CPU max MHz:                     4800.0000
CPU min MHz:                     800.0000
BogoMIPS:                        5799.77
L1d cache:                       256 KiB
L1i cache:                       256 KiB
L2 cache:                        2 MiB
L3 cache:                        16 MiB
NUMA node0 CPU(s):               0-15
Vulnerability Itlb multihit:     KVM: Mitigation: VMX unsupported
Vulnerability L1tf:              Not affected
Vulnerability Mds:               Not affected
Vulnerability Meltdown:          Not affected
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled v
                                 ia prctl and seccomp
Vulnerability Spectre v1:        Mitigation; usercopy/swapgs barriers and __user
                                  pointer sanitization
Vulnerability Spectre v2:        Mitigation; Enhanced IBRS, IBPB conditional, RS
                                 B filling
Vulnerability Srbds:             Not affected
Vulnerability Tsx async abort:   Not affected
```

- Multi-processing (多處理程序/多進程)：
  1. 資料在彼此間傳遞變得更加複雜及花時間，因為一個 process 在作業系統的管理下是無法去存取別的 process 的 memory
  2. 適合需要 CPU 密集，像是迴圈計算

3. ex. 各個工廠都分配一個員工去作事，這種方式稱作多行程 (Multi-processing) 平行執行

- Multi-threading (多執行緒/多線程)：
  1. 資料彼此傳遞簡單，因為多執行緒的 memory 之間是共用的，但也因此要避免會有 Race Condition 問題
  2. 適合需要 I/O 密集，像是爬蟲需要時間等待 request 回覆
  3. ex. 同一時間內把所有員工都派到同一家工廠去工作，此法稱做多執行緒 (Multi-threading) 平行執行

Implementation:

- Original:

```
ray@ray-ubuntu:~/operating-system/hw1$ ./original.out 100000000
testing: datasize=100000000
Integer 8 occurs 400597 in the array.
time cost = 161555us
Integer 8 occurs 400597 in the array.
time cost = 161117us
Integer 8 occurs 400597 in the array.
time cost = 162040us
Integer 8 occurs 400597 in the array.
time cost = 161231us
Integer 8 occurs 400597 in the array.
time cost = 159079us
average time cost is 161004 us
```

without using thread or process, 取5次做平均以減少誤差

- Multi process:

```
ray@ray-ubuntu:~/operating-system/hw1$ ./multi_process.out 100000000 16
testing: datasize=100000000, process=16
Integer 8 occurs 2162 times in the array
time cost for multi process is 21265 us
Integer 8 occurs 2162 times in the array
time cost for multi process is 19903 us
Integer 8 occurs 2162 times in the array
time cost for multi process is 20757 us
Integer 8 occurs 2162 times in the array
time cost for multi process is 20285 us
Integer 8 occurs 2162 times in the array
time cost for multi process is 19710 us
average time cost is 20384 us
```

multi-process, 取5次做平均以減少誤差

- Multi thread:

```
ray@ray-ubuntu:~/operating-system/hw1$ ./multi_thread.out 100000000 16
testing: datasize=100000000, thread=16
Integer 8 occurs 400720 times in the array
time cost for multi thread is 15985 us
Integer 8 occurs 400720 times in the array
time cost for multi thread is 17081 us
Integer 8 occurs 400720 times in the array
time cost for multi thread is 18492 us
Integer 8 occurs 400720 times in the array
time cost for multi thread is 17927 us
Integer 8 occurs 400720 times in the array
time cost for multi thread is 17317 us
average time cost is 17360 us
```

multi-thread, 取5次做平均以減少誤差

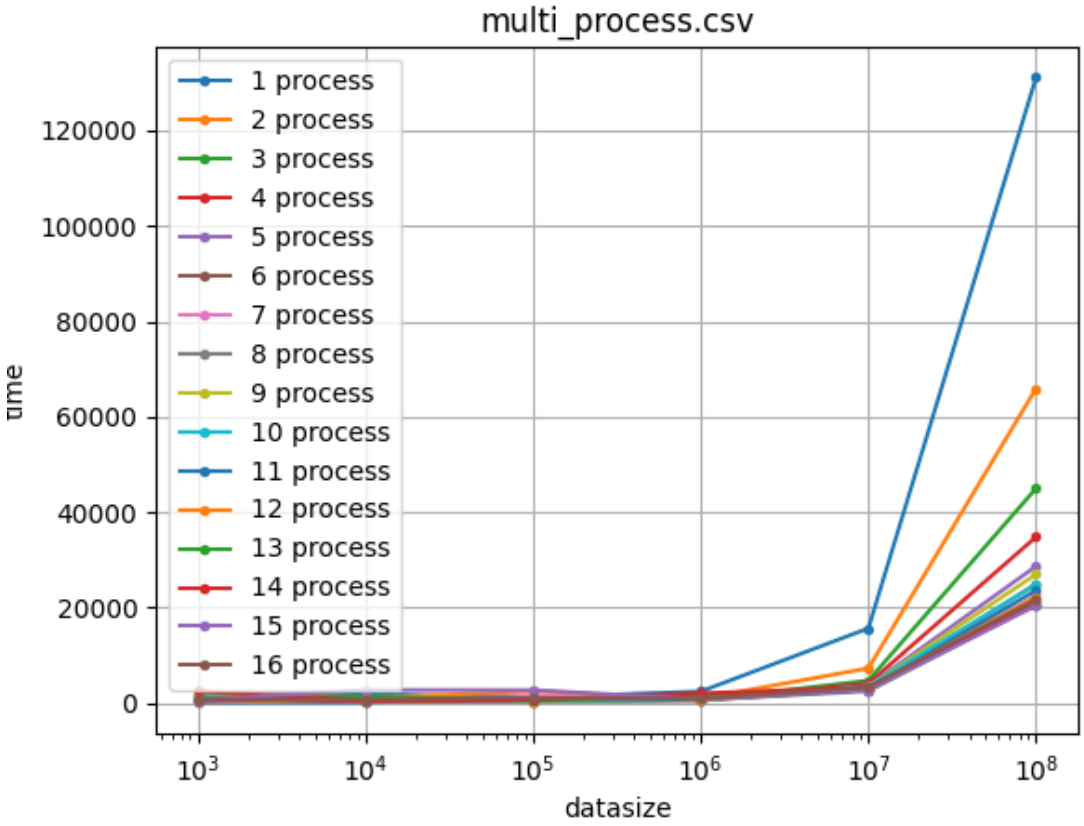
```
//MUTEX
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void* child(void* data){
    u_int64_t *input = (u_int64_t *)data;
    u_int64_t local_ans = 0; //since we only want to lock ans once
    for(u_int64_t i=input[0]; i<input[1]; i++){
        if(buffer[i]==key)
            local_ans++;
    }
    pthread_mutex_lock(&mutex); //lock
    ans += local_ans;
    pthread_mutex_unlock(&mutex); //unlock
    pthread_exit(NULL);
}
```

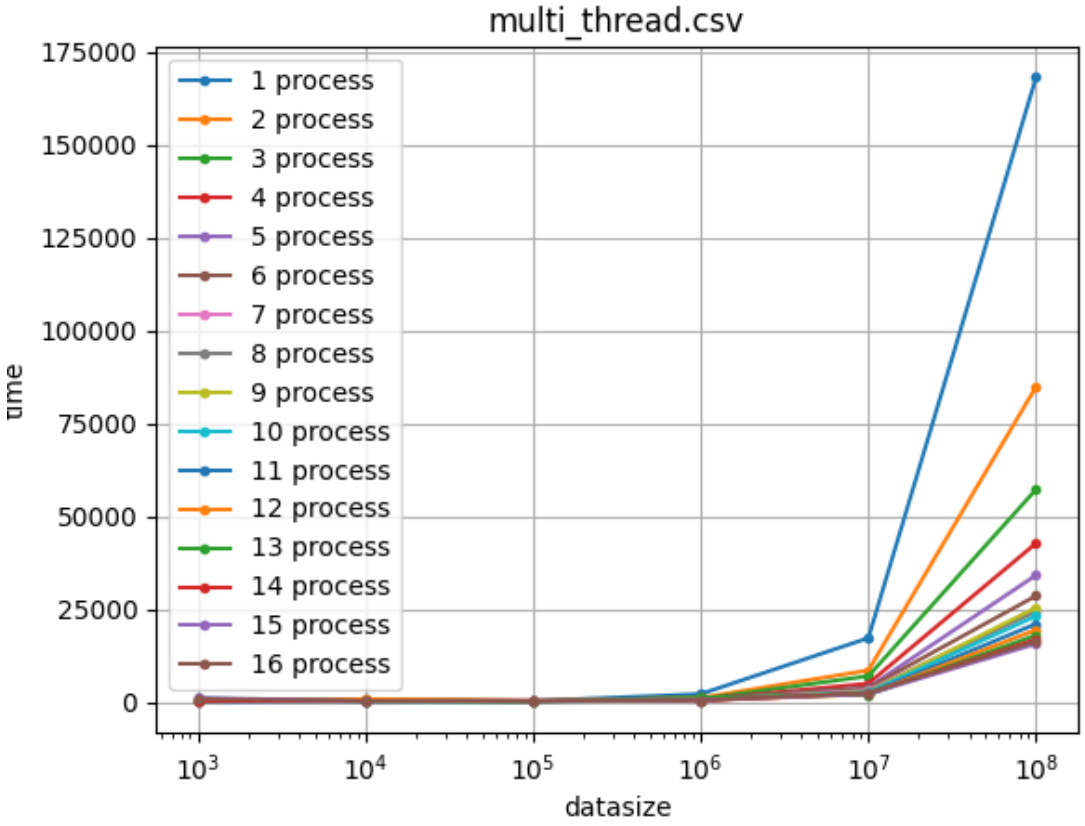
因為multi-thread記憶體是共用的，所以要加上mutex lock來鎖住他，也就是說當一個記憶體區塊被使用時，其他thread不能去碰它

Figure

- 執行時間隨data size上升而上升

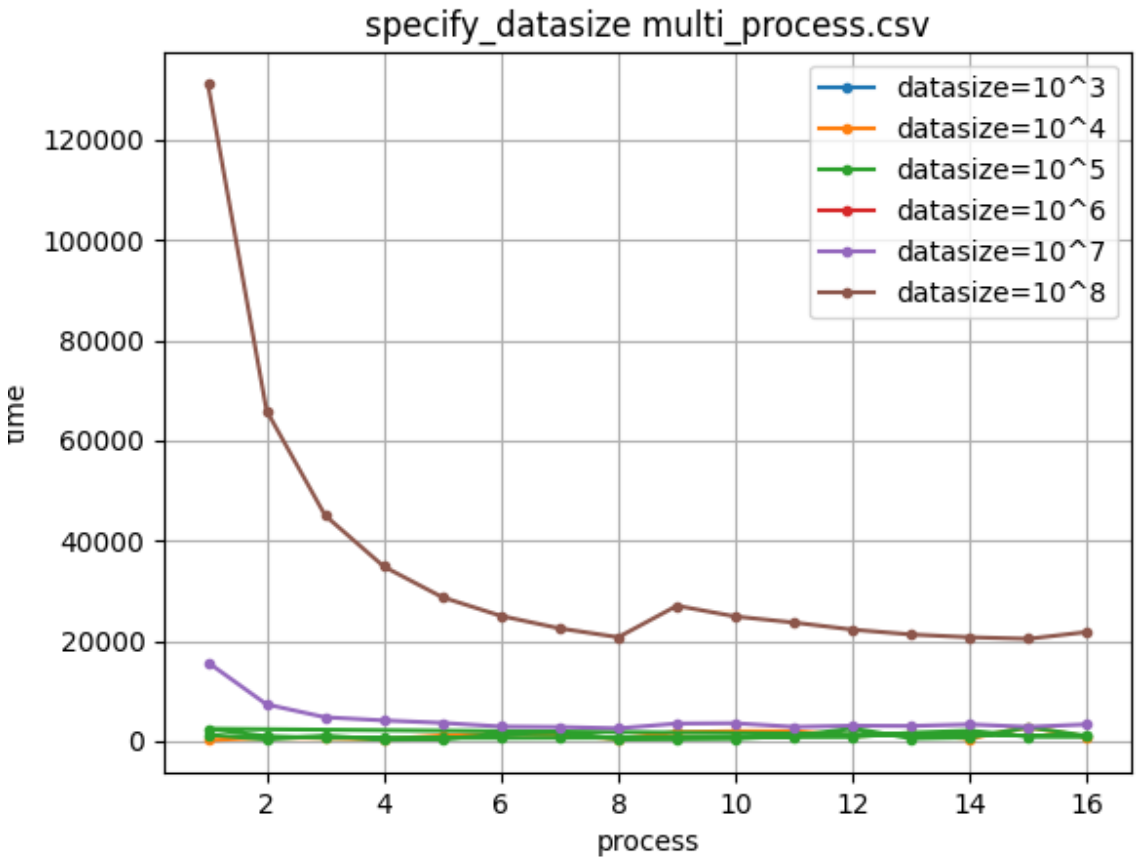
  由圖可以看出在特定process/thread(1~16)下，執行時間隨data size上升但呈非線性變化

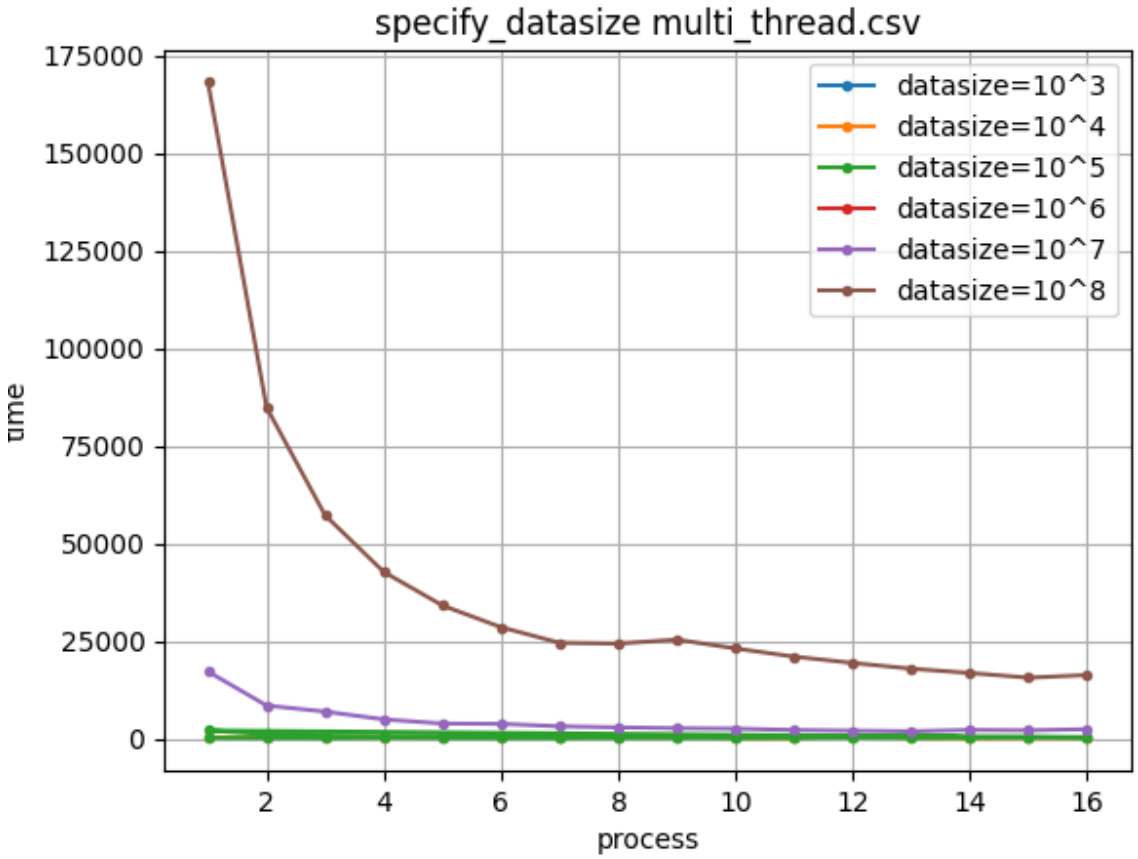

multi_process.csv
1 process
2 process
3 process
4 process
5 process
6 process
7 process
8 process
9 process
10 process
11 process
12 process
13 process
14 process
15 process
16 process
120000
100000
80000
60000
40000
20000
0
10^3
10^4
10^5
10^6
10^7
10^8
datasize



- 執行時間隨process/thread數目上升而下降

由圖可以看出在特定data size(10^3 ~ 10^8)下，執行時間隨process/thread下降但呈非線性變化


specify_datasize multi_process.csv
datasize=10^3
datasize=10^4
datasize=10^5
datasize=10^6
datasize=10^7
datasize=10^8
time
process
0
20000
40000
60000
80000
100000
120000
2
4
6
8
10
12
14
16



效能影響:

mutex對效能的影響:

mutex是一種用於多執行緒編程中，防止兩條執行緒同時對同一公共資源（比如全域變數）進行讀寫的機制，不讓multi-thread能夠同時io，也就是說，當一個記憶體區塊被使用時，其他thread不能去碰它，看是要等他讓cpu空轉還是做其他事情(這也是造成效率高或低的主因，空等太多跟別人比起來會很慢)，此外multi-process是每個process都有一份程式碼，但multi-thread則是所有thread都共用一份，所以如果程式當掉，則所有的multi-thread會全滅。

執行環境對效能的影響:

擁有較好的cpu和memory對執行效能肯定會有顯著的提升，同時因為multi process是每個process都有一份程式碼，比較吃記憶體資源，所以執行環境會對效能有蠻大的影響。