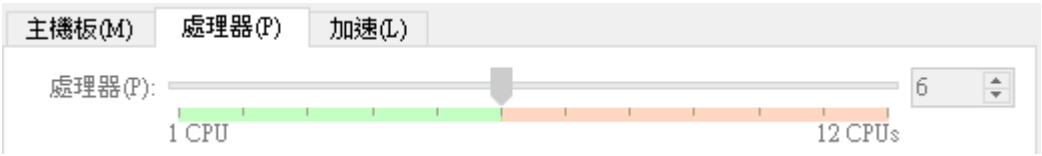
學號:E94074029
姓名:江羿賢
系級:資訊系 111 級

開發環境:
OS:Ubuntu 20.04 (使用 virtual box 虛擬機)
CPU:Intel® Core™ i7-8750H CPU @2.20GHz(使用六個執行緒,如下)



Memory:6GB
Programming Language:C++ 11 (gcc version 9.3.0)

程式執行時間:
測資大小:1.1GB
1 個 thread: 387 秒

```
please enter the number of thread 1~6 :1

start to do input
input is over, input costs 10 secs
start to do data processing
data processing is over, data processing costs 45 secs
start to do output
output is over, output costs 332 secs
program is over, total costs 387 secs
```

2 個 thread: 394 秒

```
please enter the number of thread 1~6 :2

start to do input
input is over, input costs 10 secs
start to do data processing
data processing is over, data processing costs 45 secs
start to do output
output is over, output costs 339 secs
program is over, total costs 394 secs
```

3 個 thread: 369 秒

```
please enter the number of thread 1~6 :3

start to do input
input is over, input costs 10 secs
start to do data processing
data processing is over, data processing costs 23 secs
start to do output
output is over, output costs 336 secs
program is over, total costs 369 secs
```

4 個 thread: 359 秒

```
please enter the number of thread 1~6 :4

start to do input
input is over, input costs 9 secs
start to do data processing
data processing is over, data processing costs 16 secs
start to do output
output is over, output costs 334 secs
program is over, total costs 359 secs
```

5 個 thread: 366 秒

```
please enter the number of thread 1~6 :5

start to do input
input is over, input costs 9 secs
start to do data processing
data processing is over, data processing costs 11 secs
start to do output
output is over, output costs 346 secs
program is over, total costs 366 secs
```

6 個 thread: 354 秒

```
please enter the number of thread 1~6 :6

start to do input
input is over, input costs 8 secs
start to do data processing
data processing is over, data processing costs 10 secs
start to do output
output is over, output costs 336 secs
program is over, total costs 354 secs
```

程式開發與使用說明:

先來介紹使用說明

先進行編譯:

g++ thread.cpp -std=c++11 -pthread -o thread

執行:

./thread
再來會問要使用多少執行緒:
請輸入 1~6 的其中一個數字

```
richard@richard-ubuntu-thread:~/myfile$ g++ thread.cpp -std=c++11 -pthread -o thread
richard@richard-ubuntu-thread:~/myfile$ ./thread
please enter the number of thread 1~6 :
```

這個程式總共有三個大階段 輸入 資料處理 輸出
先來講輸入的部分，利用一個 vector 當作容器，把 string 全部都接起來，而這邊要注意的是會幫每一個 string 做一個編號寫在前面，並且將前後都加上|，來使的在資料處理時比較方便以及避免亂掉。
而在資料處理的地方是使用 thread 來使迴圈達到像是平行運算的結果，在這邊會看是輸入了多少 thread，而如果輸入 1 個 thread 的話就是使用原本的 thread，就不另外多做 thread，而如果輸入 2 個 thread 的話就會外加 1 個 thread，而如果輸入 3 個 thread 的話就會外加 2 個 thread…以此類推，來幫助程式的平行程度提升，而至於在資料處理的情況下，是使用一個 vector 來存，並且依照|的位置來讀出數字，而第一個數字是位置，來使得 function 知道要把其放在哪裡，而後面其他的就是資料就把那些依照原本的順序放進去，並且依照輸入進來的參數來做出不同平行化的程度，而如果資料的數量少於 thread 的數量的話，就會有些 function 是被呼叫的，可是不會有迴圈的次數產生。
而在輸出的部分就很簡單，就按照格式，並且使用迴圈把資料處理完的資料全部印出來。

效能分析報告:
在不同的 thread 的情況下，輸入階段以及輸出階段的時間是差不多的，而會因為 thread 而減少的是在資料處理的部分，而這邊跟 memory 跟 disk I/O 比較沒有太多關係，所以會主要去分析在資料處理的過程中 CPU 的運行狀況
以下是在各 thread 中的資料處理過程的 CPU 情況，以及各階段所耗費的時間
1 個 thread

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 9569 richard   20   0 2121908   2.0g   2624 R  99.9  34.1   0:12.84 thread
```

```
please enter the number of thread 1~6 :1

start to do input
input is over, input costs 10 secs
start to do data processing
data processing is over, data processing costs 45 secs
start to do output
output is over, output costs 332 secs
program is over, total costs 387 secs
```

2 個 thread

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 9609 richard   20   0 2130104   2.0g   3292 R  99.9  34.1   0:11.71 thread
```

```
please enter the number of thread 1~6 :2

start to do input
input is over, input costs 10 secs
start to do data processing
data processing is over, data processing costs 45 secs
start to do output
output is over, output costs 339 secs
program is over, total costs 394 secs
```

3 個 thread

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 9620 richard   20   0 2138300   2.0g   3368 R  99.9  34.1   0:07.65 thread
 9621 richard   20   0 2138300   2.0g   3368 R  99.9  34.1   0:07.67 thread
```

```
please enter the number of thread 1~6 :3

start to do input
input is over, input costs 10 secs
start to do data processing
data processing is over, data processing costs 23 secs
start to do output
output is over, output costs 336 secs
program is over, total costs 369 secs
```

4 個 thread

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 9639 richard   20   0 2146496   2.0g   3268 R  99.9  34.1   0:10.20 thread
 9640 richard   20   0 2146496   2.0g   3268 R  99.9  34.1   0:10.20 thread
 9641 richard   20   0 2146496   2.0g   3268 R  99.9  34.1   0:10.19 thread
```

```
please enter the number of thread 1~6 :4

start to do input
input is over, input costs 9 secs
start to do data processing
data processing is over, data processing costs 16 secs
start to do output
output is over, output costs 334 secs
program is over, total costs 359 secs
```

5 個 thread

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 9664 richard   20   0 2154692   2.0g   3308 R  99.9  34.1   0:06.96 thread
 9665 richard   20   0 2154692   2.0g   3308 R  99.9  34.1   0:06.96 thread
 9666 richard   20   0 2154692   2.0g   3308 R  99.9  34.1   0:06.96 thread
 9663 richard   20   0 2154692   2.0g   3308 R  99.7  34.1   0:06.94 thread
```

```
please enter the number of thread 1~6 :5

start to do input
input is over, input costs 9 secs
start to do data processing
data processing is over, data processing costs 11 secs
start to do output
output is over, output costs 346 secs
program is over, total costs 366 secs
```

6 個 thread

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 9706 richard   20   0 2162888   2.0g   3276 R  99.9  34.1   0:06.87 thread
 9708 richard   20   0 2162888   2.0g   3276 R  99.9  34.1   0:06.86 thread
 9705 richard   20   0 2162888   2.0g   3276 R  99.9  34.1   0:06.86 thread
 9707 richard   20   0 2162888   2.0g   3276 R  99.9  34.1   0:06.86 thread
 9704 richard   20   0 2162888   2.0g   3276 R  99.7  34.1   0:06.85 thread
```

```
please enter the number of thread 1~6 :6

start to do input
input is over, input costs 8 secs
start to do data processing
data processing is over, data processing costs 10 secs
start to do output
output is over, output costs 336 secs
program is over, total costs 354 secs
```

在輸入以及輸出的部分，在這 6 個 thread 中都沒有太多的差異，而輸出因為要輸出很多的資料，所以花了很多時間在輸出上，又受限於 disk I/O 的寫入速度影響，所以對資料處理進行 thread 的程式應該是比較適合在輸入輸出資料量很少，但是資料處理量很大的程式來說效果會更加顯著一些。
而在資料處理的部分，memory 都是足夠用的，因為寫入的資料量相對於我的 memory 來講不算是很大的一筆資料量，而在這邊並不會對 disk I/O 讀取以及寫入，所以沒甚麼影響。而在這邊可以發現到在 1 個 thread 跟 2 個 thread 的資料處理時間是差不多的，因為在 1 個 thread 的時候，是用原本的那個 thread，而 2 個 thread 的時候，原本那個 thread 幾乎沒有在使用，而是用另外叫出來的 thread 來進行運算，所以導致時間差不多，而在 2~6 個 thread 的情況當中，原本的那個 thread 是幾乎沒有在運作的，所以有在頻繁運作的 thread 中會少一個。而在資料處理過程中，1~6 個 thread 分別花了 45,45,23,16,11,10 秒，可以看的出來在做到 3 個 thread 的時候，速度上是有感的提升了，而因為處理的資料量的關係，到 5.6 個 thread 就已經差不多了，所以在跑資料處理的迴圈時，能用多一點 thread 是比較好的，能夠讓平行度增加，但是相對的需消耗電腦的更多資源來給這支程式，而因為我自己硬體上設備的關係，只能給到 6 個

thread，要不然可以給更多 thread 可以再快一點點，而在這個過程中的 CPU 幾乎都已經到 99%的滿載狀態了，當需要處理的資料如果多又複雜的話 thread 可以帶來很好的效益。
所以總結來說，我認為作業系統是可以再做一些優化的，當程式沒有做使迴圈平行度增加時，而需處理的資料量又很大又與執行順序無關時，應該要去看 CPU 還有沒有剩下的 thread 是尚未使用的，使得該程式可以在資料處理的過程中幫忙去分配給其他 thread 來一起運算，這樣可以使得程式的執行時間更少，並且更有效率的使用 CPU。