

1.Count Sort(h5_problem1.cpp)

1.If we try to parallelize the for i loop (the outer loop), which variables should be private and which should be shared?

Ans:

Shared: **Variables a and n** must be shared, as you would need to access variable a and have access to its size. The **temp variable** would also be shared, since each array position would only be written by a single thread.

Private: **Variables i, j, and count** should be private.

2.If we parallelize the for i loop using the scoping you specified in the previous part, are there any loop-carried dependences? Explain your answer.

Ans: No, there would be no dependencies between iterations since it would not be necessary to have access to information from another iteration during code execution. Furthermore, the temp variable can be written by threads without generating concurrency problems.

3.Can we parallelize the call to memcpy? Can we modify the code so that this part of the function will be parallelizable?

Ans: No, but we can modify the code so that this part of the function can be parallelized. (The only concern would be the size of n which should not be the same. Also, the first and second arguments of the memcpy function should be pointers to elements of the array from which writing should start.)

Here's the code example:

```
#pragma omp parallel num_threads(thread_count) shared(temp, a, n)
{
    int local_n = n / thread_count;
    int index_initial = omp_get_num_thread() * local_n;
    memcpy(&a[index_initial], &temp[index_initial], local_n * (sizeof(int)));
}
```

4. Write a C Program that includes a parallel implementation of Count sort.

Ans: View h5_problem1.cpp

5. How does the performance of your parallelization of Count sort compare to serial Count sort? How does it compare to the serial qsort library function?

Ans:

Performance: $(n^2) / (\text{The number of threads})$.

But the limit of the number of threads is upper bound of n .

The qsort's performance is $O(n \log n)$.

=====

1. What have you done

在 function Count_sort 裡 for 迴圈的地方，我把它用 openmp 做平行化，但是 count 這個變數需要設為 private，讓每個 thread 對變數 count 都有各自的副本，才會得到預期的結果，然後變數 i 跟 j 我直接宣告在 for 迴圈的地方，就不用額外設成 private。

排序 10 個數字，1 個 thread，結果正確：

```
H54084078@pn1:~> ./h5_problem1 1
Please input how many number you want to sort: 10
Input the sequence you want to sort: 55 44 21 33 98 63 20 0 78 855
0 20 21 33 44 55 63 78 98 855
The execution time = 0.000000
H54084078@pn1:~> |
```

排序 10 個數字，8 個 thread，結果正確：

```
H54084078@pn1:~> ./h5_problem1 8
Please input how many number you want to sort: 10
Input the sequence you want to sort: 55 44 21 33 98 63 20 0 78 855
0 20 21 33 44 55 63 78 98 855
The execution time = 0.000000
H54084078@pn1:~> |
```

排序 10 個數字，16 個 thread，結果正確：

```
H54084078@pn1:~> ./h5_problem1 16
Please input how many number you want to sort: 10
Input the sequence you want to sort: 55 44 21 33 98 63 20 0 78 855
0 20 21 33 44 55 63 78 98 855
The execution time = 0.000000
H54084078@pn1:~> |
```

2. Analysis on your result

這題需要 sort 非常多數字，才能看出單一 thread 跟 multi thread 執

行時間的差別，因此我排序 20,000 個數字，利用單一 thread 跟 500

個 thread 進行比較。

1 個 thread:

[illegible]

500:

[illegible]

thread 設成 16，consumer thread 設定成 $34 - 16 = 18$ 個。

然後我定義了 5 個 function：

(1)insert_queue：把從 article folder 裡的 txt 檔讀到的每一行當作一個 node 插入到 single share queue 的尾端，而且在 insert 的過程，我用 #pragma omp critical 保護起來

(2)del_queue：從 single share queue 的頭端取出一個 node，取出 node 的過程我用 #pragma omp critical 保護起來

(3)tokenize：把 single share queue 裡面的每個 node 做 strtok(把每一行裡的 keyword 拆開)，如果拆開後得到的 keyword 是我有興趣的關鍵字，我就把那個關鍵字的數量加 1，關鍵字的數量我開一個 array 叫 num_of_key 儲存起來。

(4)readfile：一行一行的讀取 article folder 裡的 txt 檔，並呼叫 insert_queue 做插入到 single share queue 的動作


(5)process：分別處理 producer 跟 consumer。producer 呼叫 readfile，consumer 呼叫 del_queue 和 tokenize

然後在全部動作執行完後會返回 main function 做印出 keyword 跟數量的動作，在印之前我用 #pragma omp barrier 做同步

2. Analysis on your result

我要讀的檔案都放在 article 資料夾裡，裡面我放了 16 個檔案，內容我都放一樣比較好檢查有沒有寫錯。

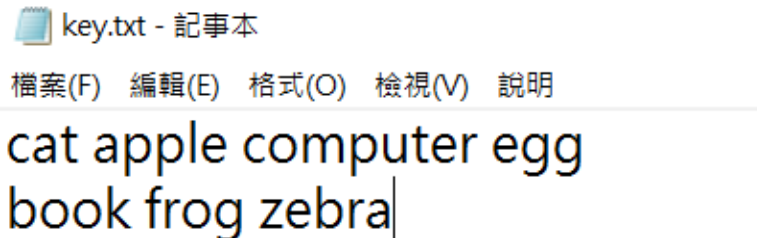
article folder 裡的 txt 檔(16 份一模一樣)：



a1 (1).txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明
Apple book cat dog
Egg frog green hat
ice jacket

key.txt 放關鍵字

key.txt:



key.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明
cat apple computer egg
book frog zebra

分析 thread num 大於，等於，小於 article 數量兩倍(32)，執行時間差異(因為我預設是把 thread num 平均分配給 producer 跟 consumer。若 thread num > article 數量的兩倍時，producer 數量我就會設定成等於 article 數量，consumer 數量等於 thread num – producer)

Thread num = 2 vs 8 vs 16 vs 32 vs 64 vs 128

2:

```
H54084078@pn1:~> ./h5_problem2 2
```

Our Keyword : cat apple computer egg book frog zebra

[illegible]

-Keyword Count Summary-

```
cat:16
apple:16
computer:0
egg:16
book:16
frog:16
zebra:0
```

-Keyword Count Summary-

Total time = 0.026270

H54084078@pn1:~>



8:

```
Windows PowerShell h54084078@LAI
H54084078@pn1:~> ./h5_problem2 8

Our Keyword : cat apple computer egg book frog zebra

Thread 3 : Apple book cat dog
Thread 2 : Apple book cat dog
Thread 3 : Egg frog green hat
Thread 3 : ice jacket
Thread 0 : Apple book cat dog
Thread 0 : Egg frog green hat
Thread 0 : ice jacket
Thread 2 : Egg frog green hat
Thread 2 : ice jacket
Thread 3 : Apple book cat dog
Thread 0 : Apple book cat dog
Thread 3 : Egg frog green hat
Thread 3 : ice jacket
Thread 2 : Apple book cat dog
Thread 2 : Egg frog green hat
Thread 0 : Egg frog green hat
Thread 2 : ice jacket
Thread 0 : ice jacket
Thread 3 : Apple book cat dog
Thread 3 : Egg frog green hat
Thread 3 : ice jacket
Thread 2 : Apple book cat dog
Thread 2 : Egg frog green hat
Thread 2 : ice jacket
Thread 0 : Apple book cat dog
Thread 0 : Egg frog green hat
Thread 0 : ice jacket
Thread 2 : Apple book cat dog
Thread 2 : Egg frog green hat
Thread 3 : Apple book cat dog
Thread 3 : Egg frog green hat
Thread 3 : ice jacket
Thread 0 : Apple book cat dog
Thread 0 : Egg frog green hat
Thread 0 : ice jacket
Thread 2 : ice jacket
Thread 1 : Apple book cat dog
Thread 1 : Egg frog green hat
Thread 1 : ice jacket
Thread 1 : Apple book cat dog
Thread 1 : Egg frog green hat
Thread 1 : ice jacket
Thread 1 : Apple book cat dog
Thread 1 : Egg frog green hat
Thread 1 : ice jacket
Thread 1 : Apple book cat dog
Thread 1 : Egg frog green hat
Thread 1 : ice jacket

-----Keyword Count Summary-----
cat:16
apple:16
computer:0
egg:16
book:16
frog:16
zebra:0
-----Keyword Count Summary-----
Total time = 0.047243
H54084078@pn1:~> |
```

Windows 在 這裡輸入文字來搜尋

16:

```
Windows PowerShell  X  h54084078@LAP
Total time = 0.047243
H54084078@pn1:~> ./h5_problem2 16

Our Keyword : cat apple computer egg book frog zebra

Thread 1 : Apple book cat dog
Thread 1 : Egg frog green hat
Thread 1 : ice jacket
Thread 6 : Apple book cat dog
Thread 5 : Apple book cat dog
Thread 2 : Apple book cat dog
Thread 3 : Apple book cat dog
Thread 3 : Egg frog green hat
Thread 3 : ice jacket
Thread 7 : Apple book cat dog
Thread 7 : Egg frog green hat
Thread 7 : ice jacket
Thread 3 : Apple book cat dog
Thread 3 : Egg frog green hat
Thread 6 : Egg frog green hat
Thread 1 : Apple book cat dog
Thread 3 : ice jacket
Thread 1 : Egg frog green hat
Thread 1 : ice jacket
Thread 4 : Apple book cat dog
Thread 4 : Egg frog green hat
Thread 4 : ice jacket
Thread 0 : Apple book cat dog
Thread 0 : Egg frog green hat
Thread 0 : ice jacket
Thread 7 : Apple book cat dog
Thread 7 : Egg frog green hat
Thread 7 : ice jacket
Thread 5 : Egg frog green hat
Thread 4 : Apple book cat dog
Thread 4 : Egg frog green hat
Thread 4 : ice jacket
Thread 0 : Apple book cat dog
Thread 0 : Egg frog green hat
Thread 0 : ice jacket
Thread 2 : Egg frog green hat
Thread 2 : ice jacket
Thread 5 : ice jacket
Thread 6 : ice jacket
Thread 2 : Apple book cat dog
Thread 2 : Egg frog green hat
Thread 2 : ice jacket
Thread 6 : Apple book cat dog
Thread 6 : Egg frog green hat
Thread 6 : ice jacket
Thread 5 : Apple book cat dog
Thread 5 : Egg frog green hat
Thread 5 : ice jacket
-----Keyword Count Summary-----
cat:16
apple:16
computer:0
egg:16
book:16
frog:16
zebra:0
-----Keyword Count Summary-----
Total time = 0.039714
H54084078@pn1:~>
```



在這裡輸入文字來搜尋

32:

```
Windows PowerShell h54084078@LAP
Total time = 0.039714
H54084078@pn1:~> ./h5_problem2 32

Our Keyword : cat apple computer egg book frog zebra

Thread 4 : Apple book cat dog
Thread 4 : Egg frog green hat
Thread 3 : Apple book cat dog
Thread 4 : ice jacket
Thread 12 : Apple book cat dog
Thread 3 : Egg frog green hat
Thread 3 : ice jacket
Thread 2 : Apple book cat dog
Thread 2 : Egg frog green hat
Thread 2 : ice jacket
Thread 1 : Apple book cat dog
Thread 1 : Egg frog green hat
Thread 9 : Apple book cat dog
Thread 11 : Apple book cat dog
Thread 8 : Apple book cat dog
Thread 9 : Egg frog green hat
Thread 1 : ice jacket
Thread 5 : Apple book cat dog
Thread 5 : Egg frog green hat
Thread 5 : ice jacket
Thread 0 : Apple book cat dog
Thread 0 : Egg frog green hat
Thread 0 : ice jacket
Thread 10 : Apple book cat dog
Thread 10 : Egg frog green hat
Thread 10 : ice jacket
Thread 6 : Apple book cat dog
Thread 7 : Apple book cat dog
Thread 7 : Egg frog green hat
Thread 7 : ice jacket
Thread 6 : Egg frog green hat
Thread 12 : Egg frog green hat
Thread 9 : ice jacket
Thread 15 : Apple book cat dog
Thread 15 : Egg frog green hat
Thread 15 : ice jacket
Thread 11 : Egg frog green hat
Thread 11 : ice jacket
Thread 6 : ice jacket
Thread 13 : Apple book cat dog
Thread 13 : Egg frog green hat
Thread 8 : Egg frog green hat
Thread 8 : ice jacket
Thread 12 : ice jacket
Thread 14 : Apple book cat dog
Thread 14 : Egg frog green hat
Thread 14 : ice jacket
Thread 13 : ice jacket

-----Keyword Count Summary-----
cat:16
apple:16
computer:0
egg:16
book:16
frog:16
zebra:0
-----Keyword Count Summary-----
Total time = 0.075932
H54084078@pn1:~> |
```



在這裡輸入文字來搜尋

64:

```
Windows PowerShell  X  h540840

Segmentation fault
H54084078@pn1:~> ./h5_problem2 64

Our Keyword : cat apple computer egg book frog zebra

Thread 1 : Apple book cat dog
Thread 1 : Egg frog green hat
Thread 1 : ice jacket
Thread 2 : Apple book cat dog
Thread 2 : Egg frog green hat
Thread 2 : ice jacket
Thread 6 : Apple book cat dog
Thread 7 : Apple book cat dog
Thread 6 : Egg frog green hat
Thread 7 : Egg frog green hat
Thread 7 : ice jacket
Thread 6 : ice jacket
Thread 0 : Apple book cat dog
Thread 8 : Apple book cat dog
Thread 8 : Egg frog green hat
Thread 8 : ice jacket
Thread 3 : Apple book cat dog
Thread 3 : Egg frog green hat
Thread 3 : ice jacket
Thread 9 : Apple book cat dog
Thread 5 : Apple book cat dog
Thread 9 : Egg frog green hat
Thread 9 : ice jacket
Thread 11 : Apple book cat dog
Thread 4 : Apple book cat dog
Thread 10 : Apple book cat dog
Thread 11 : Egg frog green hat
Thread 11 : ice jacket
Thread 14 : Apple book cat dog
Thread 14 : Egg frog green hat
Thread 0 : Egg frog green hat
Thread 10 : Egg frog green hat
Thread 10 : ice jacket
Thread 12 : Apple book cat dog
Thread 12 : Egg frog green hat
Thread 0 : ice jacket
Thread 14 : ice jacket
Thread 4 : Egg frog green hat
Thread 15 : Apple book cat dog
Thread 15 : Egg frog green hat
Thread 15 : ice jacket
Thread 13 : Apple book cat dog
Thread 13 : Egg frog green hat
Thread 13 : ice jacket
Thread 12 : ice jacket
Thread 5 : Egg frog green hat
Thread 5 : ice jacket
Thread 4 : ice jacket

-----Keyword Count Summary-----
cat:16
apple:16
computer:0
egg:16
book:16
frog:16
zebra:0

-----Keyword Count Summary-----
Total time = 0.188538
H54084078@pn1:~> |
```



在這裡輸入文字來搜尋

128:

```
Windows PowerShell h54084078
Total time = 0.188538
H54084078@pn1:~> ./h5_problem2 128

Our Keyword : cat apple computer egg book frog zebra

Thread 2 : Apple book cat dog
Thread 2 : Egg frog green hat
Thread 2 : ice jacket
Thread 1 : Apple book cat dog
Thread 3 : Apple book cat dog
Thread 3 : Egg frog green hat
Thread 3 : ice jacket
Thread 5 : Apple book cat dog
Thread 6 : Apple book cat dog
Thread 6 : Egg frog green hat
Thread 6 : ice jacket
Thread 7 : Apple book cat dog
Thread 8 : Apple book cat dog
Thread 9 : Apple book cat dog
Thread 10 : Apple book cat dog
Thread 11 : Apple book cat dog
Thread 12 : Apple book cat dog
Thread 13 : Apple book cat dog
Thread 14 : Apple book cat dog
Thread 15 : Apple book cat dog
Thread 5 : Egg frog green hat
Thread 5 : ice jacket
Thread 7 : Egg frog green hat
Thread 7 : ice jacket
Thread 8 : Egg frog green hat
Thread 8 : ice jacket
Thread 9 : Egg frog green hat
Thread 9 : ice jacket
Thread 10 : Egg frog green hat
Thread 10 : ice jacket
Thread 11 : Egg frog green hat
Thread 11 : ice jacket
Thread 12 : Egg frog green hat
Thread 12 : ice jacket
Thread 13 : Egg frog green hat
Thread 13 : ice jacket
Thread 14 : Egg frog green hat
Thread 14 : ice jacket
Thread 15 : Egg frog green hat
Thread 15 : ice jacket
Thread 1 : Egg frog green hat
Thread 1 : ice jacket
Thread 4 : Apple book cat dog
Thread 4 : Egg frog green hat
Thread 4 : ice jacket
Thread 0 : Apple book cat dog
Thread 0 : Egg frog green hat
Thread 0 : ice jacket
-----Keyword Count Summary-----
cat:16
apple:16
computer:0
egg:16
book:16
frog:16
zebra:0
-----Keyword Count Summary-----
Total time = 0.420810
H54084078@pn1:~>
```



在這裡輸入文字來搜尋

2 thread < 16 thread < 8 thread < 32 thread < 64 thread < 128 thread 的

執行時間。推論出以下兩點：

(1)結果顯示此題 create thread 的 overhead 蠻大的，只要多 create 一些 thread，執行時間就增加。

(2)推論出只有一個 producer 跟一個 consumer 的執行速度最快。

若要多個 producer 跟多個 consumer 的話，producer 數量為 article 數量的一半時(每個 producer 負責兩個 article)，執行速度會比

producer 數量為 article 數量的 1/4 倍(每個 producer 負責四個 article)來的快。(此部分 create thread 的 overhead 並不大)

然而這題因為會牽涉到比較複雜的 insert、delete、search，所以很多個 thread 執行的話 overhead 會比第一題的 count sort 來的大，執行時間會較長(32 thread < 64 thread < 128 thread 的執行時間)

3.Any difficulties?

花了不少時間在研究讀資料夾裡面全部的檔案

4.(optional) Feedback to TAs

^o^