多處理機平行程式設計 2021 fall 作業五說明

題目

1. Count Sort

Count sort is a simple serial sorting algorithm that can be implemented as follows:

```
void Count_sort(int a[], int n) {
    int i, j, count;
    int* temp = malloc(n∗sizeof(int));
    for (i = 0; i < n; i++) {
        count = 0;
        for (j = 0; j < n; j++)
            if (a[j] < a[i]) count++;
        else if (a[j] == a[i] && j < i)
            count++;
        temp[count] = a[i];
    }
    memcpy(a, temp, n∗sizeof(int));
    free(temp);
} /∗ Count sort ∗/
```

The basic idea is that for each element `a[i]` in the list `a`, we count the number of elements in the list that are less than `a[i]`. Then we insert `a[i]` into a temporary list using the subscript determined by the count. There's a slight problem with this approach when the list contains equal elements, since they could get assigned to the same slot in the temporary list. The code deals with this by incrementing the count for equal elements on the basis of the subscripts. If both `a[i] == a[j]` and `j <i`, then we count `a[j]` as being "less than" `a[i]`.
After the algorithm has completed, we overwrite the original array with the temporary array using the string library function memcpy.

Please answer the following questions:

1. If we try to parallelize the for i loop (the outer loop), which variables should be private and which should be shared?
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.
3. Can we parallelize the call to `memcpy` ? Can we modify the code so that this part of the function will be parallelizable?
4. Write a C program that includes a parallel implementation of Count sort.
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?

2. Producer-Consumer

Given a keyword file that contains many keywords. Use **OpenMP** to implement a producer-consumer program in which some of the threads are producers and others are consumers. The producers read text from a collection of files, one per producer. They insert lines of text into a single shared queue **(please implement your own thread safe queue)**$^{\mathbf{NEW}}$. The consumers take the lines of text and tokenize them. **Tokens are “words” separated by white space**. When a consumer finds a token that is keyword, the keyword count increases one. **Please print each keyword and its count**.

> 💡 You can define your own input files. Please clearly describe how you read those files in the report.

如何交作業

1. 把作業交到你的家目錄底下 (`mv yourfile.c ~`) 並使用正確的檔名: `h5_problem1.c` (or `.cpp`) , `h5_problem2.c` (or `.cpp`) 以及第二題的文字檔，想用資料夾稍作整理也可以，但是請確保只有一個 `h5_problem1.c` 和一個 `h5_problem2.c` 。 請勿抄襲。
2. 上傳你的 report (使用 `.md` 或是 `.pdf` 格式，或是包含連結至你的 HackMD 頁面 / GitHub Readme 的文字檔 $^{\mathrm{NEW}}$)(in Chinese or English) 至 moodle 並包含：
   - What have you done
   - Analysis on your result
   - Any difficulties?
   - (optional) Feedback to TAs

截止日期： 2021/12/24 23:59:59

Please report any server mis-configuration you found. TAs are new to System/Network administration. We will appreciate your report.

另外，pn1 上已安裝 `perf` ，有在 pn1 上測試的同學可以試試看。NEW

計分方式

- 程式 style 25%
  1. 巢狀結構需用階層式編排。
  2. 適當地使用空白列來區隔功能上無關的程式碼，使你的程式段落分明。
  3. 清楚且詳細的註解。
- 結果正確無誤 20%
  平行程式執行的結果需與循序程式執行的結果一致。
- 效能 30%
- 平行程式的觀察報告 25%
  請針對作業中提到的問題逐一回答，相對應的觀察及分析請寫在報告中，