**OS HW1 Document**

1. 開發環境

在主機上使用虛擬環境開發

- 主機配備: Intel® Core™ I5-9400H CPU @ 2.90GHz, 32G RAM

- 虛擬機: 6核心處理器、4G記憶體

- 作業系統: Linux 64bit (Ubuntu 22.04)

- 程式語言: C++11

1. 實作方法與流程

主程式流程:

* 1. 取得使用者輸入(檔名、方法數、切割份數)
  2. 根據使用者方法數(若方法數為1則不需要切割份數)
  3. 計時開始
  4. 執行排序
  5. 計時結束
  6. 寫檔
  7. 詢問是否繼續? 是->回到步驟1. 否->結束

氣泡排序:

1. 外迴圈選定最尾端未排序元素
2. 內迴圈從頭開始與相鄰元素比較，若靠前元素較小則交換，直到比較至外迴圈選定元素

合併:

1. 複製左陣列及右陣列元素至兩個佇列內
2. 比較兩個佇列排頭之元素，將較小者取出，並放入大陣列
3. 重複步驟2.，直至兩個佇列元素為空為止

方法一(直接氣泡排序):

1. 呼叫氣泡排序函式做排序

方法二(切割成K份氣泡排序後合併):

1. 將N份資料平均切割成K份，算出其每筆資料區間的index值
2. 將K份資料及其開始、結束index值依序傳入氣泡排序函式做排序
3. 將K份資料，兩兩傳入合併排序函式做合併
4. 令K值為合併後份數，並且檢查K值 大於1->回到步驟3. 等於1->結束

方法三(多處理元對K份資料氣泡排序後合併):

1. 建立一份儲存N筆資料的共享記憶體(供多處理元之間使用)
2. 將N份資料平均切割成K份，算出其每筆資料區間的index值
3. 建立K個處理元，每個處理元將一份資料傳入氣泡排序函式做排序
4. 建立K-1個處理元，每個處理元將兩份資料傳入合併排序函式做合併
5. 令K值為合併後份數，檢查K? 大於1->回到步驟4. 等於1->結束

在步驟3.及步驟4.結束處，Parent Process皆會使用waitpid()來等待每個Fork出去的Child Process返回，以確保在執行下一步驟後，其資料不會應順序錯亂而出錯。

方法四(多執行緒對K份資料氣泡排序後合併):

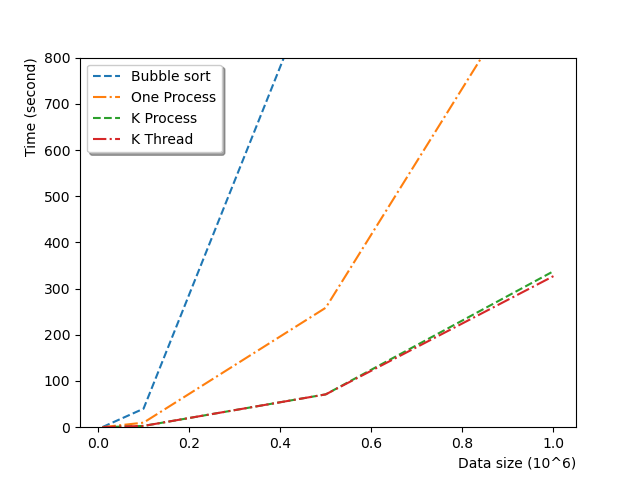
1. 將N份資料平均切割成K份，算出其每筆資料區間的index值
2. 建立K個執行緒，每個執行緒將一份資料傳入氣泡排序函式做排序
3. 建立K-1個執行緒，每個執行緒將兩份資料傳入合併排序函式做合併
4. 令K值為合併後份數，檢查K? 大於1->回到步驟3. 等於1->結束

與方法三相同，在步驟2.及步驟3.結束處，會使用pthread\_join()函式來等待每個建立出的執行緒返回，再進行下個步驟。

1. 分析結果和原因

實驗一(K = 4)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1萬筆資料 | 10萬筆資料 | 50萬筆資料 | 100萬筆資料 |
| 方法一 | 0.496秒 | 43.034秒 | 1002.5秒 | 3667.83秒 |
| 方法二 | 0.11秒 | 9.95秒 | 242.532秒 | 1003.8秒 |
| 方法三 | 0.03秒 | 0.561秒 | 62.013秒 | 335.233秒 |
| 方法四 | 0.032秒 | 0.574秒 | 62.01秒 | 313.682秒 |



由圖表可以看出，執行效率最差的是方法一，其次為方法二，方法三及方法四效率則為最佳，且效率相當。

氣泡排序的執行效率為O(n^2)，因此單次執行時間花費約正比於n^2，

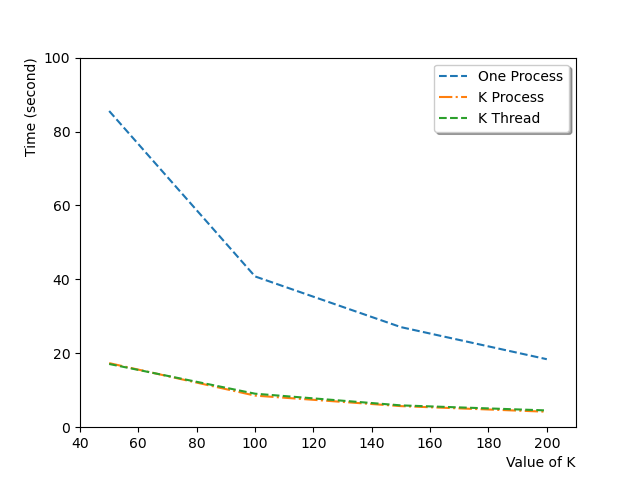
所以方法二約快於方法一執行時間的4倍，符合實驗結果。

而方法三及方法四，由於使用了多處理元及多執行緒，4個排序工作是同步並行處理的，因此其實際執行時間則約需要再除以4，理論上方法三、方法四效率會快於方法二4倍時間、快於方法一16倍時間。

而實驗中方法三及方法四也確實快於方法二接近4倍，而這缺少的一小部份時間，應該是由於方法三在做CPU排程中切換時，需要耗費較多時間去做Context Switch；而方法四則是因為User Thread在CPU中的排程僅是一個單位，因此無法如同方法三一次分配到多個排程。

實驗二(N = 1,000,000)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | K = 50 | K = 100 | K = 150 | K = 200 |
| 方法一 | 3667.83秒 | 3667.83秒 | 3667.83秒 | 3667.83秒 |
| 方法二 | 63.418秒 | 34.407秒 | 27.006秒 | 17.3秒 |
| 方法三 | 16.218秒 | 8.438秒 | 5.301秒 | 4.102秒 |
| 方法四 | 16.183秒 | 8.907秒 | 5.832秒 | 4.313秒 |



由圖表可以看出，執行效率最差的還是方法一，而方法二、三、四當K值愈大，效率愈佳。

合併單輪K段所有資料的效率約為O(n)，將不同的資料份數分別代入後，以k = 50和100舉例，得出K = 50執行時間約為K = 100的2倍，與實驗結果相符，而當K值慢慢增大後，其2倍差距會愈來愈不明顯，因其函數呼叫、參數傳遞會愈多次，程式語言需要基本的處理時間來處理這些排序或合併以外之行為。極端狀況下，K = N，就等於純合併排序，這種狀況下，系統還要花時間去建立Process或Thread就為了對只有一個元素的陣列做氣泡排序，因此效率絕對不會是最好的。

1. 問題與心得

Window環境下不能使用<sys/wait.h>等標頭文件

因windows不支持fork()的系統調用，也不支持waitpid()等函數

解決方法

在不熟悉window API的相應函數下，只好使用虛擬機在linux環境下做，但由於不常使用虛擬機，在工具使用上也有許多不便

計時不準確問題:

原先使用C++較舊版本所提供的ctime函式庫，應該是clock()函式測量時間的方法是僅測量該Process所使用的CPU周期數，而未算到所建立出去的Process處理時間，才造成如此結果。

解決方式:

使用C++11所提供的時間函式庫 – chrono