作業系統

HW1-書面報告

班級:資訊三甲

學號:**11027142**

姓名:林蕙郁

- ## 開發環境

  利用Python程式碼以及Python 3.11.5-64bit的Interpreter實作。

- ## 實作方法和流程

  讀檔案:用split line的方式去得到每個資料。

  寫檔案:去判斷輸出檔案是否存在，沒有的話創新檔案，有的話重新寫入。

  把排序過後的資料、CPU執行時間、以及寫檔案的時間(+8時區)寫入。

  排序的時候利用import numpy及from numba import jit做加速。

  任務一:直接利用氣泡排序把N筆資料排完，並將資料存到list型別內。

  任務二:先決定好chunk_size(利用取ceiling去確保每筆資料都能儲存在某個chunk裡面)，然後將chunk_size筆資料做氣泡排序後存入sorted_chunk。

  當所有資料都在sorted_chunk後，再利用合併排序去把資料做合併，存入merge_chunk，且每次都以兩份sorted_chunk的資料做排序並放。最後所有資料合併完之後會存入sorted_chunk之第0個位置，即為我們要寫入檔案之結果。

  任務三:先決定好chunk_size，且利用Queue來做process和process的溝通。利用p = Process(target=parallel_bubble_sort, args=(data[start_index:end_index], q))去選擇我所要執行的目標函數、傳入之參數、最後結果之存放位置，當要求都做好之後就開始執行process。去queue中獲取排序之後的chunk存入sorted_chunk以便之後做合併排序。然後利用p.join()去等待所有process結束，獲得經過氣泡排序過後的所有sorted_chunk。接下來要將sorted_chunk的資料利用合併排序獲取最終結果，和上述一樣，利用Queue、Process去實作，直到將資料排序過後存入sorted_chunk[0]，這個位置之資料就是我們要寫入檔案之資料。

  任務四:和任務三很像，但是是利用thread去實作，可以節省時間、增進效率。利用t = threading.Thread(target=bubble_sort_thread,

args=(data[start_index:end_index], sorted_chunks, i))使用執行緒實作，一樣要實作執行緒，也要開始執行執行緒，也需要等到所有執行緒結束才能算是完整地完成一次排序。最終也是將最終結果存入sorted_chunks[0]，並且傳到寫檔案的function內將資料內容寫入檔案。

- 探討結果和原因

  實驗結果紀錄：

K={10,20}	N=1萬	N=10萬	N=50萬	N=100萬
方法一	444.974, X	20926.086, X	492413.792, X	1844933.253, X
方法二	332.037, 325.005	1745.181, 981.022	43212.221, 21191.031	186092.254, 91890.612
方法三	5652.346, 8555.443	10629.014, 21204.722	18201.311, 25141.338	45553.400, 33598.658
方法四	380.002, 394.013	1841.215, 1082.205	44401.267, 22200.323	186527.738, 91535.413

單位：ms

(備註：方法一之K=20時沒有執行時間是因為方法一並不用切K份，所以以該CPU執行時間作為實驗記錄)

表1數據:

不同**N && K=10**	Task 1	Task 2	Task 3	Task 4
N=10000	444.974	332.037	5652.346	380.002
N=100000	20926.086	1745.181	10629.014	1841.215
N=500000	492413.792	43212.221	18201.311	44401.267
N=1000000	1844933.253	186092.254	45553.4	186527.738

單位:毫秒

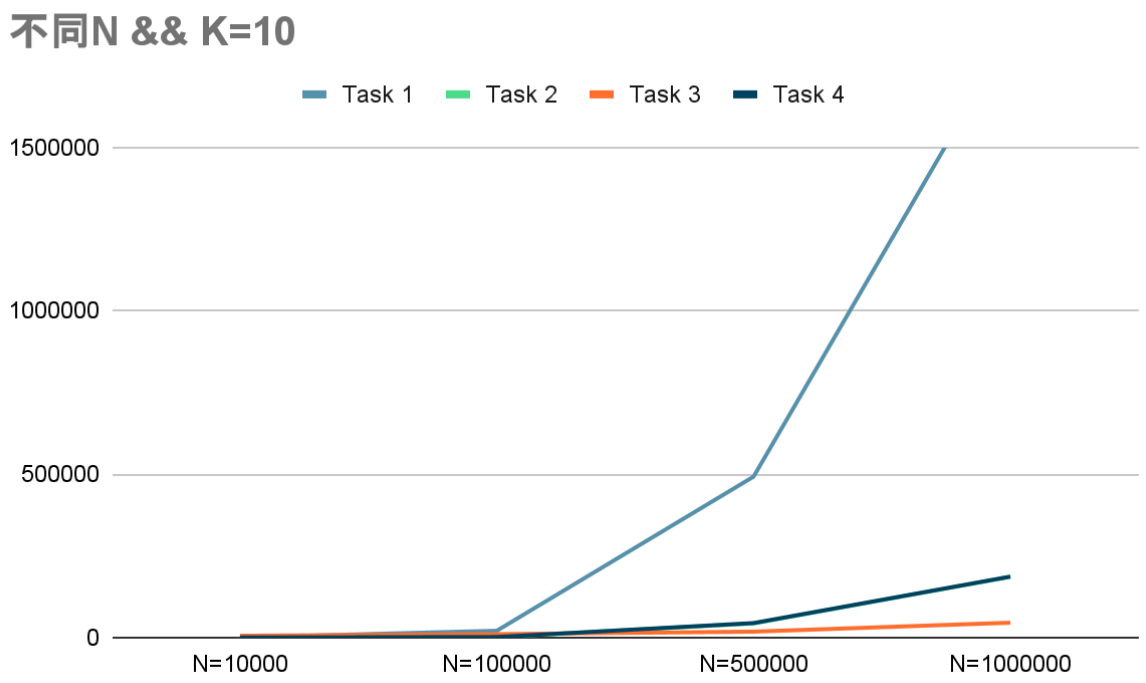

表1:不同N值, K=10, 其x軸為N值, y軸為執行時間(單位:毫秒)

備註:task2及task4所耗費時間相近, 所以在表中較難看出差異。

導致結果之原因:

Task1所耗費時間比其他來的慢是因為資料筆數過大的時候, 程式並不能分開處理數據, 他必須要一整份數據下去處理, 所以當數據資料量超大的時候, 效率會極低。

Task3所耗費時間比Task4還要少是因為Python在執行執行緒有GIL(全局解釋器鎖)之限制, 會限制執行緒一次只能執行一個, 然而process卻可以同時平行執行, 所以任務3之速度會比4還要來的快一點。

表2數據:

N=50萬 && 不同K值	Task 1	Task 2	Task 3	Task 4
K = 10	492413.792	43212.221	18201.311	44401.267
K = 20	492431.792	21191.031	25141.338	22200.323

單位:毫秒

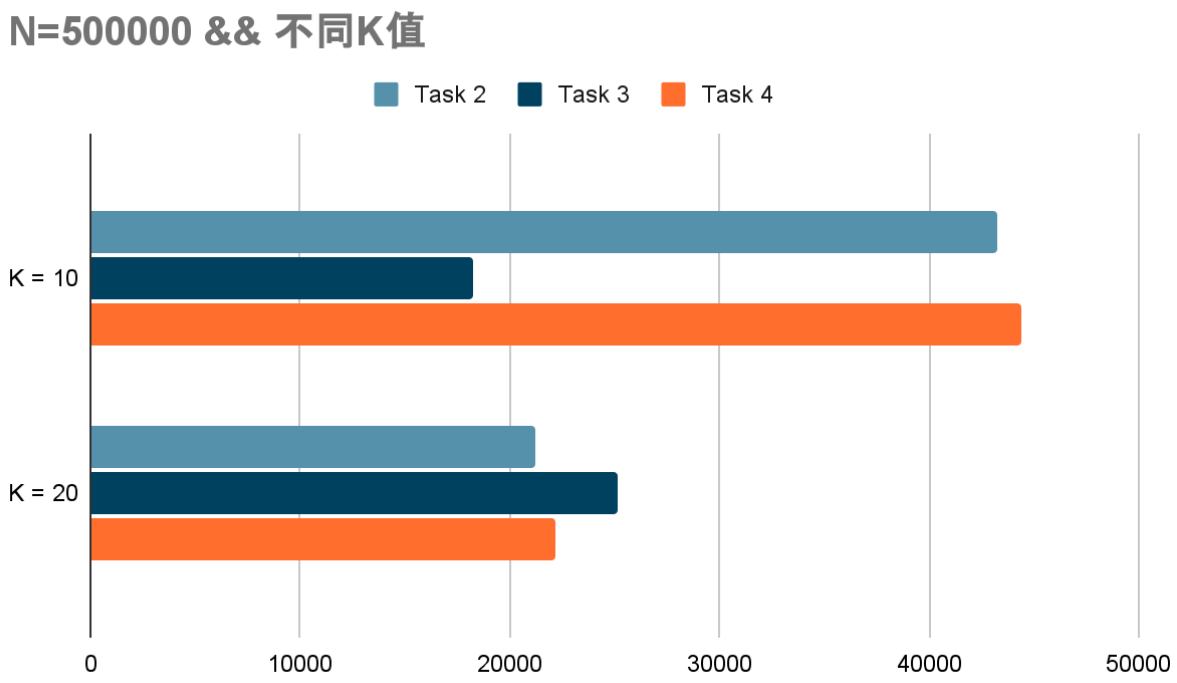

表2:不同K值, N=50萬, 其x軸為執行時間(單位:毫秒)

備註:這張圖片並未包含task1, 因為想多探討及觀察task2~4之所耗時間, 如果加入task1會無法看清其他task之差別, 且task1也用不到K值。

導致結果之原因:

Task2之效率有差是因為chunk數量不同, 且氣泡排序跟合併排序之效率成長並非線性, 切的數量越多, 氣泡排序效率變好(資料量不大), 但合併排序的效率變差(因為要耗費更多次的合併), 但因為兩種的big O不相同, 所以並不會有抵銷(指排序時間增減之抵銷)。

Task3是因為利用多process平行執行, 且不受GIL之限制, 所以確定可以多process同時運作, 增加效能, 和其他兩個任務筆起來的話有明顯的效率增加。

Task4是因為GIL導致不能真正的多thread一起執行, 所以對於效率的提升較不顯著。