OS Project1書面報告

資訊三甲 11027164 趙怡儒

1. 開發環境 : Visual Studio Code 使用Python
2. 實作方法和流程 :

方法一：將input存成list，直接BubbleSort，並輸出結果以及cpu time及日期時間。

方法二：將input存成list後，將資料分成k份，將每一份先自己BubbleSort後，再兩個兩個MergeSort，最後兩個merge完即排列完成，輸出結果以及cpu time及日期時間。

方法三：將input存成list後，將資料分成k份，由K個processes各別進行BubbleSort，再用K-1個process(es)作MergeSort，排列完成後輸出結果以及cpu time及日期時間。

方法四：與任務三流程相同，不同的地方在於方法四是由K個threads各別進行BubbleSort，再用K-1個thread(s)作MergeSort，排列完成後輸出結果以及cpu time及日期時間。

在執行方法三時，一開始發現執行速度偏慢，在想應該是process的資料要傳輸的話需要另外消耗時間，找了網路上的方法，找到在 Python 的 multiprocessing 模組中，Manager類提供了一種創建共享對象的方法。當使用 Manager創建Queue時，返回的是一個Queue，這個Queue是在多個process之間共享的，可以進行跨process通信，且為了讓function都可以正常使用，將每個方法主要的資料結構都改成這種類型的Queue。

1. 探討結果和原因 ：

1.

一張含有 文字, 螢幕擷取畫面, 行, 繪圖 的圖片

自動產生的描述

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | N=1w | N=10w | N = 30w | N=50w | N=100w |
| 方法一 | 3.905 | 475.735 | 4978.695 | X | X |
| 方法二 | 0.252 | 12.615 | 64.503 | 177.027 | 1903.266 |
| 方法三 | 2.529 | 15.469 | 15.193 | 35.890 | 185.940 |
| 方法四 | 0.264 | 7.309 | 65.510 | 327.918 | 764.266 |

表一不同N值的各個方法執行時間(K值固定為50)

由圖與表格可以得知，方法一因為單純用BubbleSort所以明顯慢其他方法非常多，甚至50萬與100萬的資料執行太久所以改成用30萬呈現他效率慢的程度，方法二的部分則是減少了每次BubbleSort排序的資料量，所以效率也快了很多，再來是方法三和方法四，看圖跟表格可以發現誰比較有效率跟資料量的大小有關，10萬前方法四比較有效率，自己在私底下也有試過1000筆資料，也是方法四明顯快很多，10萬後卻反過來，我認為應該跟我有用Manager的queue有關係，但不確定。

2.

一張含有 文字, 螢幕擷取畫面, 行, 繪圖 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | K = 10 | K = 50 | K = 100 | K = 1000 |
| 方法二 | 0.515 | 0.252 | 0.240 | 0.290 |
| 方法三 | 1.398 | 2.529 | 4.466 | 39.506 |
| 方法四 | 0.526 | 0.264 | 0.290 | 0.799 |

表二不同K值的各個方法執行時間(N值固定1w)

因為方法一不受k值影響不做討論，方法二明顯K越大越快，因為BubbleSort一次跑的數據量減少，而方法三會隨著K越大效率越慢，推測是當process越多時，會需要耗費更多時間做context switch，因此效率明顯變差。方法四效率都很快應該就是因為多個thread不僅可達到同時運算，又能共用地址空間。

3.

一張含有 文字, 行, 螢幕擷取畫面, 繪圖 的圖片

自動產生的描述

一張含有 文字, 字型, 螢幕擷取畫面, 數字 的圖片

自動產生的描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | K = 10 | K = 50 | K = 100 | K = 1000 |
| 方法二 | 36.669 | 12.615 | 3.712 | 0.813 |
| 方法三 | 8.104 | 15.469 | 13.123 | 48.169 |
| 方法四 | 36.291 | 7.309 | 3.824 | 1.265 |

表三不同K值的各個方法執行時間(N值固定10w)

結論與第二點大致相同，但因為資料量十分龐大，thread同時運算，又共用地址空間的優勢大幅上升，所以K值越大效率越好的情況非常明顯，同時觀察了工作管理員在執行方法三時，處理程序會變多，CPU使用率也會變高。