Parallel Programming Homework 1 Report

112062524 CS23 詹博允

1. Implementation
    - How do you handle an arbitrary number of input items and processes?
        - 我的做法是將其input item 跟 process 相除後取 ceiling, 有想過其他方式, 但是在 coding 上 ceiling 的 code 最不會在 edge case 上出問題, 所以在後續 debug 上就改成用 ceiling。

    - How do you sort in your program?

      我將 sorting 的部分分成兩個區塊:每個 process 內部的 sorting 跟 odd-even 的 sorting.

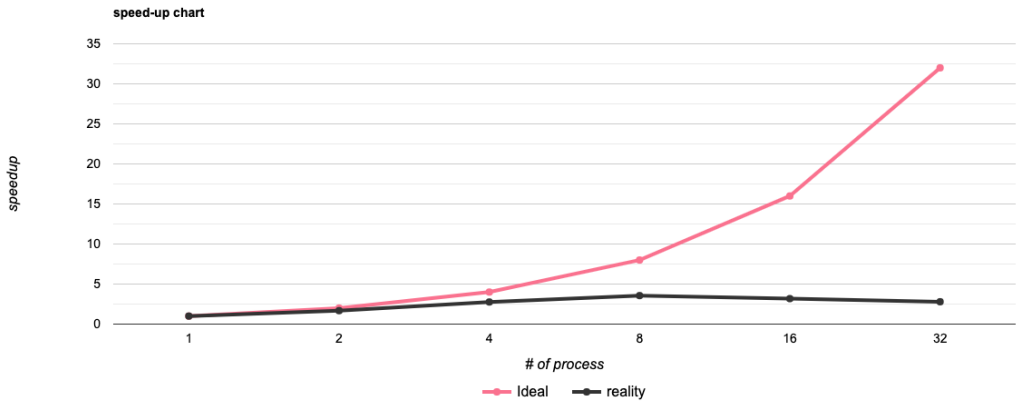
        - 內部 sorting : 為了提升效率, 本來在這個步驟使用了兩種不同的 sorting 方式, 如果該 process 分配到的數量 < 16 則會用 insertion sort, 反之則會用 qsort。但是因為發現 < 16 的 case 實在是不, 所以後來全部改成了qsort。後來為了要提升速度, 又將其改成std::sort, 整體的速度也有顯著的提升。

        - Odd-even sorting : 我使用的方法是用 merge sort, 維護三個 index, 分別跑過兩個 array, 以此來形成第三個 array的資料, 不過有做一些額外的維護, 像是判斷如果兩個陣列的順序已經確定不變, 則會直接 return , 以此來提升效率。

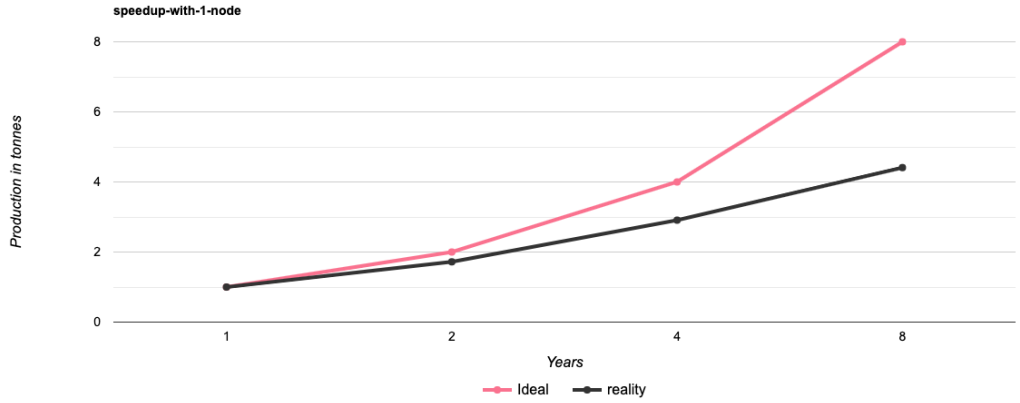
    - Other efforts you've made in your program.
        - 有參考了網路上別人的做法, 在一些 edge case 做剪枝, 像是先不要把整個 array 傳過來, 而是先傳一個值過來確認大小後再將整個 array 傳過來, 以及像是前面提到的將 qsort改成sort以及將資料

2. Experiment & Analysis
    a. Methodology
        - **我的實驗都是在這堂課提供的 cluster 上實作, 我主要是用 open testcase 29 來做測試, baseline 則是用 1個node與1個 process去做運算**
        - **測量方法:基本上都是用 MPI_Wtime() 在不同的地方做時間節點後去相減得到數值, 然後去做比較, 以及觀測 IPM profiler 所提供的資料來討論**
    b. Plots: Speedup Factor & Time Profile

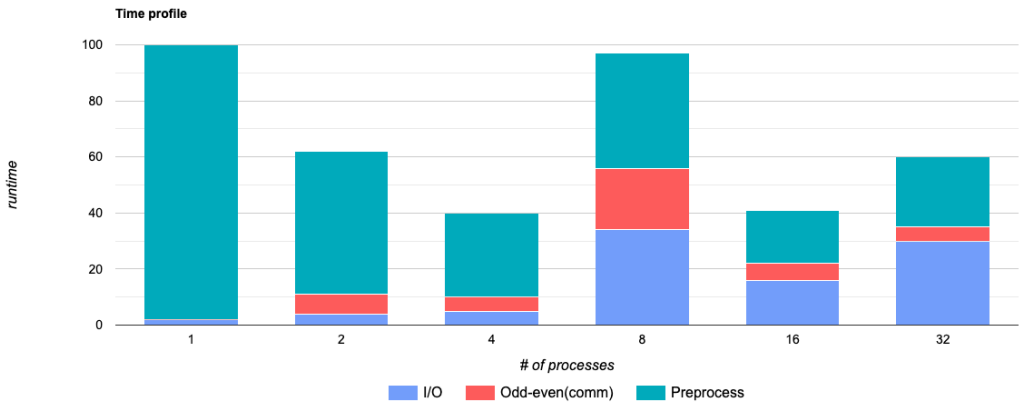
- 第一個實驗我測試了在不同的 process下的 speedup，我使用了 1, 2, 4, 8, 16, 32 個 process 去跑 testcase 29，測試方法則是用 MPI_Wtime()函式在 MPI_init()之後 跟MPI_Finalize之前，並取各個process的最大值。



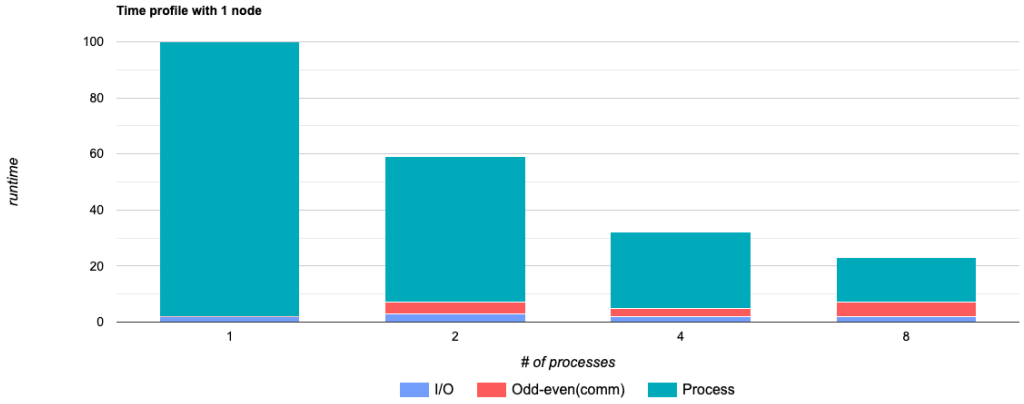
- 上圖中的粉線是理想的 speed-up , 黑線則是 reality speed-up
- 第二個實驗則是在1個node的前提下跑 1, 2 ,4,8 個 process 去測試 speed-up



- 下圖則是 time profile, 一樣是 1,2,4,8,16,32 個 process 去跑 testcase 29,



- 第二個實驗一樣是用1個node 跑 1,2,4,8個 process 去跑 tesetcase 29



c. Discussion & optimization

- Scale-up rate並不如預期：理想上的speed-up應該要跟投入的 node與process成等比成長，但是事實上的 scale-up 並不是這樣，而是卡在2-3倍，可以理解成是因為 communication overhead 形成的bottleneck，這點也跟老師上課時提到得內容相符合, 理 send/recv function, performance drop, server

- 在單一 node時的scale-up比multi-node還好：可以看到單一node的整體時間減少比multi-node還漂亮，我認為主要原因是在 communication overheada倍最小化了，所以其整體成長才能這麼好看

3. Experiences & Conclusion

- Your conclusion of this assignment.

- 我覺得這次的 hw 主要的困難點在於大量資料下的debug, 因為平型化程式的debug流程比起一般的程式還要複雜許多, 我有很多的bug都是莫名其妙的就解掉了, 所以我覺得我需要更多的練習。另外我也發現在排行榜上很多人的排名都比我前面, 但是這次礙於時間壓力沒有時

間做更多的優化, 另外我也發現在大型資料上用不同的方法優化時間會有顯著的提升, 這點也是我感到蠻訝異的

- What have you learned from this assignment?
  - 我學到的主要有兩點:
    - 一是在平行化上的程式需要注意的事情比一般的程式還多, 像是可能會導致 deadlock, 或是 bottleneck在哪裡也比一般程式還難分析。
    - 二是在程式優化上, 其實有很多的優化是以往不會特別注意到的, 像是對陣列的宣告或是剪枝的行為, 所以算是學到蠻多東西
- What difficulties did you encounter in this assignment?
  - 我遇到的問題主要都是自己低能把最外圍的loop的 termination condition 寫錯導致我的時間複雜度直接變O(n^2), 其餘的話則是像是在分配數值給不同的 node 時可能會導致 segmenation fault, 我覺得很挫折的點是在很多時候我還是沒有搞懂為什麼我會在這些情況 runtime error, 我都是想說“不然把這個改改看好了”結果測資就過了, 所以很多的 bug 我到現在還在思考到底為什麼會出現
- If you have any feedback, please write it here. Such as comments for improving the spec of this assignment, etc.
  - 我覺得這個 project 本身已經十分完善了, 特別是助教還寫了很多環境變數能讓我們直接呼叫, 已經是一個十分友善的 project 了, 十分感謝助教的幫忙