Report

1.黃柏惟

110164508

2. Implementation

一開始我會先判斷可用的processes以及 要處理的input數量。 如果input數量比process少的話我就free掉一些process讓input數量等於process數量。 如果processes的數量比input還要多的話就平均分配 如果除不盡的話就把多的分給前面幾個。 並且在分配完後使用boost 的spread sort進行第一次的排序。

在odd even sort階段，會分成odd phase跟even phase去進行討論。 每次輪到的phase就會將他所有的data傳給rank–1的process。 由於兩方都是sorting過的data因此可以一個一個比較會比merge起來sort還要快些。 Merge and sort完成後 rank – 1 的process就會將後半段的data傳回去給rank 並完成該次sorting。 此動作會一直重複到當所有的階段都完成排序後結束。

3. Experiment & Analysis

a. system spec: 使用課程提供的apollo.cs.nthu.edu.tw

b. performance metrics: 我使用MPI\_Wtime()去把對應的操作包起來。

Figure 1

Figure2

Discussion

從Figure1可以看到當processes的數量增加的時候可以有效的縮減IO time跟 CPU time 但是只會稍微的增加一點點COMM time 所以可以看到平行化的處理是成功的。 不過在processes的數量為8的時候可以看到Comm time幾乎已經快要占了好一大部分執行時間了，如果可以在把processes的數量往上提升 e.g. 32 可以推測如果再將processes增加下去整個執行的時間都會花在processes彼此之間傳遞資料的時間上。 我想這是我的bottle neck。

4. Experiences / Conclusion

算是第一次寫平行程式，一開始花了不少時間在理解平行化的概念。開始實作時也遇到了很多次Send/Receive沒有對起來使整份code卡死的情況發生。不過在完成後可以發現整體的執行時間有因為平行化的處理而降低不少，蠻有成就感的。