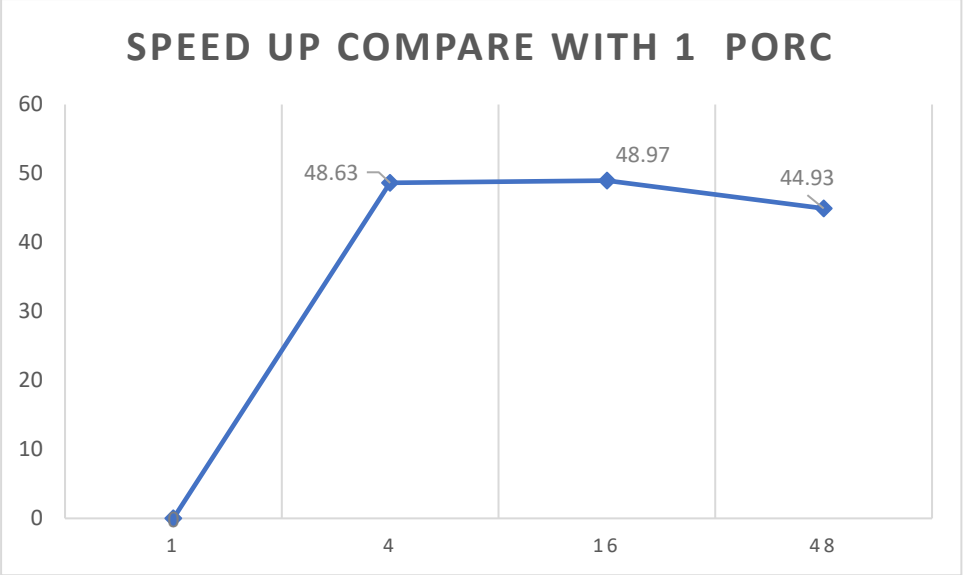
CS542200 Parallel Programming
Homework 1: Odd-Even Sort Report

106062210 陳則翰

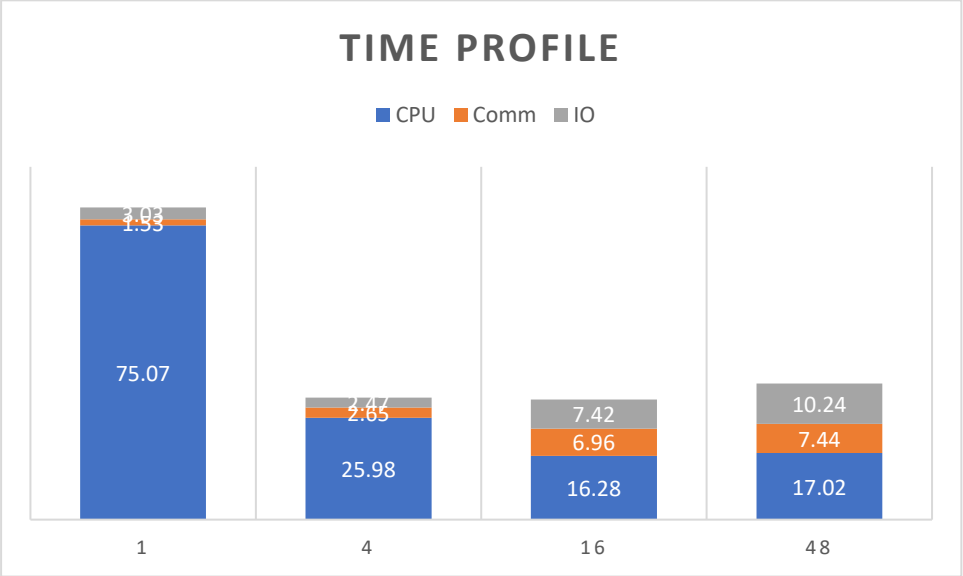
- ## Implementation

  本次作業實作 Odd-Even Sort，我先將所有 n 個 elements 平分給所有 MPI Process，如果有不能平分的情況就把多餘的 k 個分給首 k 個 Process。之後，給每個 process 定一個 odd_rank 跟一個 even_rank，來決定他們在每一個 phase 要跟哪個 Process 溝通。之後在 sort 過程中，首先讓每個 process 先把 local 的 element 做一次 std::sort()(只有一開始需要 因為之後都是 sorted sub array)，然後讓相鄰的兩個 process 做一次 merge_sort 裡 sort 的動作。如果只把一邊的檔案送過去做 merge 的話就只有一個 process 在工作，為了提升 work load 並且避免算完的東西還要送回去，我直接將兩邊的 array 都互相傳送。讓他們各自 merge 完之後把自己應該拿的部分放進本來的 local 位置裡。之後就照著 bubble sort 的 sorting principle: sort 到沒有改變為止。

- ## Experiments



註:testcase38



- Discussion

根據在圖表裡所能看到的，當 process 數量上升時，Scalability 簡直奇差無比，可以說只有當 process 從 1 個提昇至 4 個時有速度上的提升。至於會這樣的理由，很明顯地，雖然提升 process 的數量的確有助於需要運用到計算的部分有顯著的成長，但所帶來的 IO 時間跟 Comm 時間成長實在是太顯著

了。這邊我推測是因為當我們的 process 數成長，我們在一輪內所需要作的 SendRecv 次數也相對成長，自然 Comm 時間就會成長，如果想要改善這個問題，可以使用更聰明的演算法，例如，當我的 rank 在另外一個 phase 有改變時才會進行 SendRecv 的比較(但是如果要知道另外一個 rank 有沒有改變就又需要用另一輪 SendRecv 來作，也許不會有提升)。另一方面，不知道為什麼 IO 時間在 process 數上升時也跟著上升了，而且上升幅度還比 Comm 時間來得大。

- Difficulties

  這次 HW 中我遇到一個不可期的問題，雖然在 coding 的時候有查到當我需要同時 send 跟 recv 東西時，使用 Sendrecv()會好於分別使用 Send()跟 Recv()。但我沒有深究也沒有實際避免這麼做。結果當我的 node 數增加時(我本來都跑 node 不多的側資)，就有 process 莫名其妙收不到 Message 的情況發生，一把 Send(),Recv()改成 Sendrecv()之後就沒事了，實在是很莫名其妙，大概就是 functiony 在我看不到的地方的優化造成的這種現象吧。