You are asked to parallelize the sequential Mandelbrot Set program, and learn the following skills:

- Get familiar with thread programming using Pthread and OpenMP.
- Combine process and thread to implement a hybrid parallelism solution .
- Understand the importance of load balance.

Implementation

Briefly describe your implementation in diagrams, figures, sentences, especially in the following aspects:

✓ How you implement each of requested versions, especially for the hybrid parallelism? How do you partition the task?

首先，我原本想要靜態去依高度切割出平等圖片區塊給我的程式們去跑,但遇到下列問題,可能某些區塊要計算很久,因為 while 迴圈 iters 跑到底，然後如果跨 process 的話 image 的 access 容易打架，就算規劃好該 access 區域，底層硬體可能會不依你的意思多 access 到別的位子，故會 runtime error。

如果沒遇到這問題也可能超時，如 hybird 就算最後 mpi_gather 也會超時，故我不管 hybird 或者 pthread 都是要 dynamic allocated 高度，每次高度 1，process 或者 thread 誰先做完就 allocate 下一個高度，動態平衡 working time。

✓ **What technique do you use to reduce execution time and increase scalability?**

我 pthread 使用經典的 pthread pool 我大概把我覺需要的部分 implement 出來，就一個 producer threadpool_add 跟 consumer threadpool_thread，主要是 consumer 要等 task 時要設 condition 去 wait，這樣他就會 dynamic allocate input 接下來因為是同 process 下的 thread 故只要把同個 global 變數把它適時 lock 住,他就可以跑得好。

hybird 部分我犧牲一 個 rank process 當作主 process 他負責去動態分配高度給其他 process 去 compute 那一高度圖片資訊，在接收 work process 的資料，把他寫入 image 這樣就不會不同 process 打架，但會少一個 process compute。時間上就是動態平衡去減少不必要的等待增加 **scalability**。

✓ Other efforts you’ve made in your program.

1.我 hybird 有試著要再用 pthread 去讓主 process 去做計算但就像我所說如果動態 allocate 給主 process work by thread 會出現跟其他 process 在硬體層級打架，但如果 static 部分給 pthread，則要等所有其他 process 做好再回傳，理論上是可行，但是我實作遇到 time out 之類的問題，可能我自己沒寫好，故放棄。也有用 pthread 版的 pool 但是不知為何 consumer 無法把 queue head 的資量讀出來，故也失敗。

2.openmp 盡量可以去整理好 for loop 中變數去讓他可以最有效平行化。

3.mpi Isend 再 Recv 來減少主 process wait 時間，因為誰做完就分配給誰，故不用擔心誰會錯過新的工作訂單。

1. Experiment & Analysis

Explain how and why you do these experiments? Explain how you collect those measurements? Show the result of your experiments in plots, and explain your observations.

i、 Methodology

(a). **System Spec** (If you run your experiments on your own machine)

apollo again~

(b). **Performance Metrics**

How do you measure computing time of your programs? How do you compute the values in the plots?
for both I use slow01 as the setting because its height is 1439 (large) and its iteration 174170376(can let it have some extreme long computatoin case). In addition, it is beautiful XD and makes me feel ㄎㄤ

for hybird:

I try to just compute the time of computing pixels for each rank, and master rank 0 will compute the time of controling communication and wait to write images (that means rank 0 will wait other rank done, so its time is the whole time)

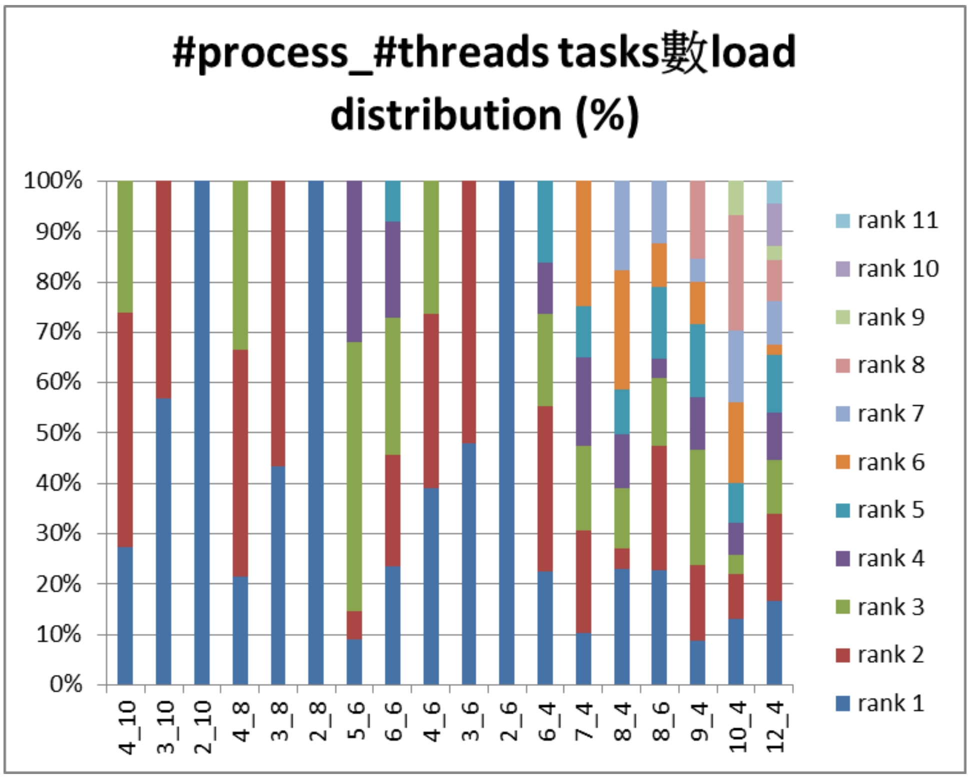
Also I look for the number of task distribution on each process.

for pthread:

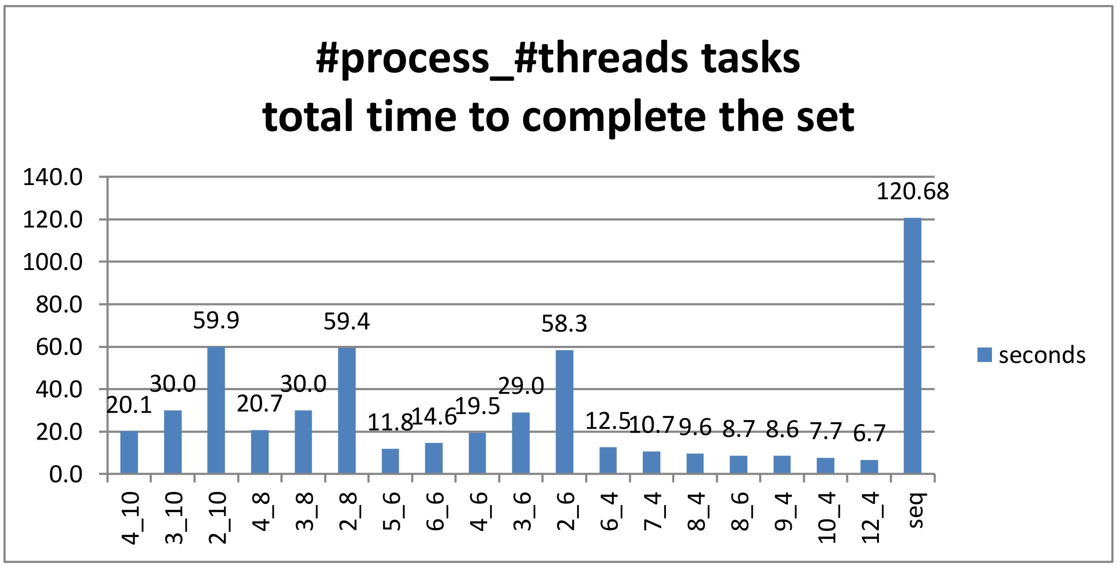
I try to just compute the average time of computing pixels for each thread. Also I look for the number of task distribution on each process.

ii、 Plots: Scalability & Load Balancing

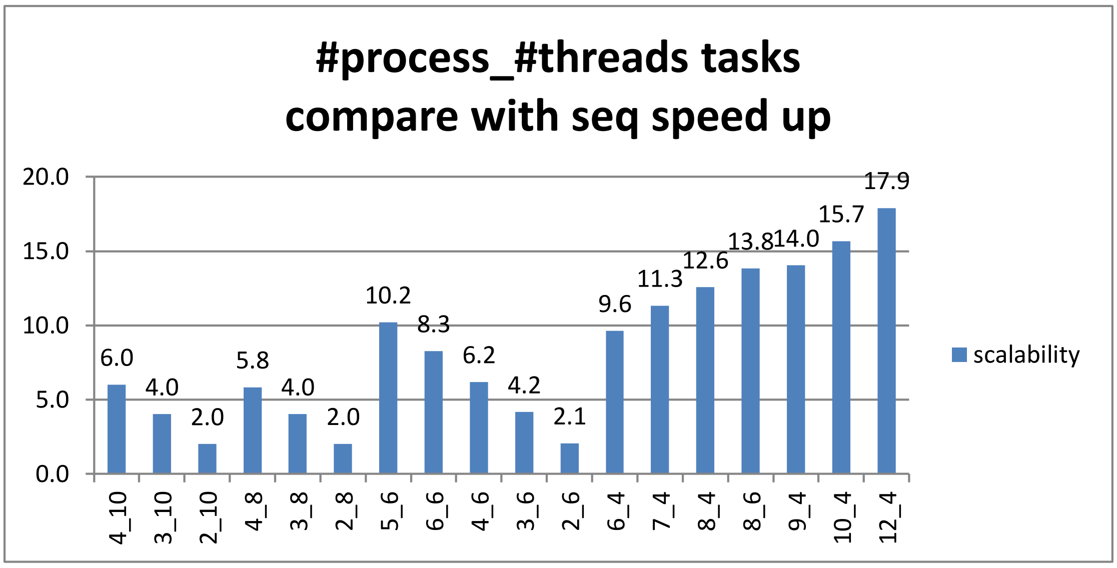
(a). Conduct **strong scalability** experiments, and plot the speedup. The plot must contain at least 4 different scales (number of processes, threads) for both single node and multi-node environments.



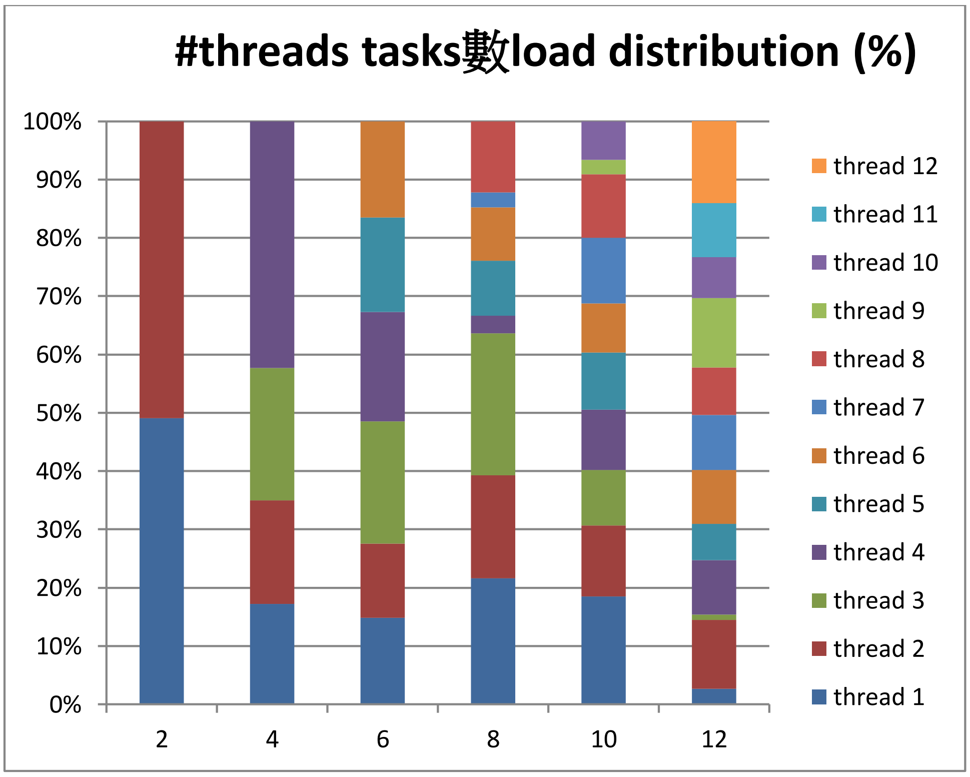
(Figure.1 **#process_#threads tasks 數 load distribution (%)** for hybird master do not compute pixels)



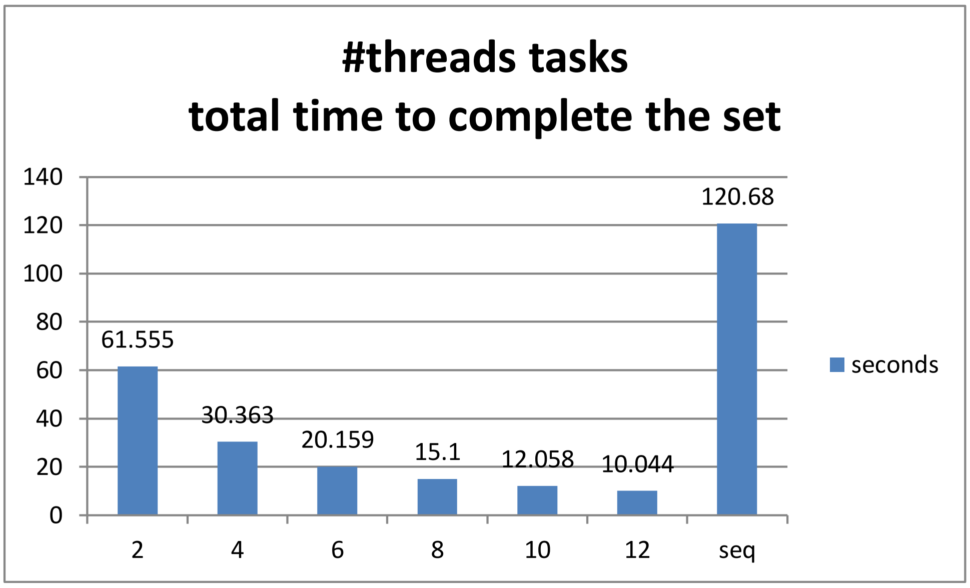
(Figure.2 **#process_#threads total time for seconds** for hybird master do not compute pixels)



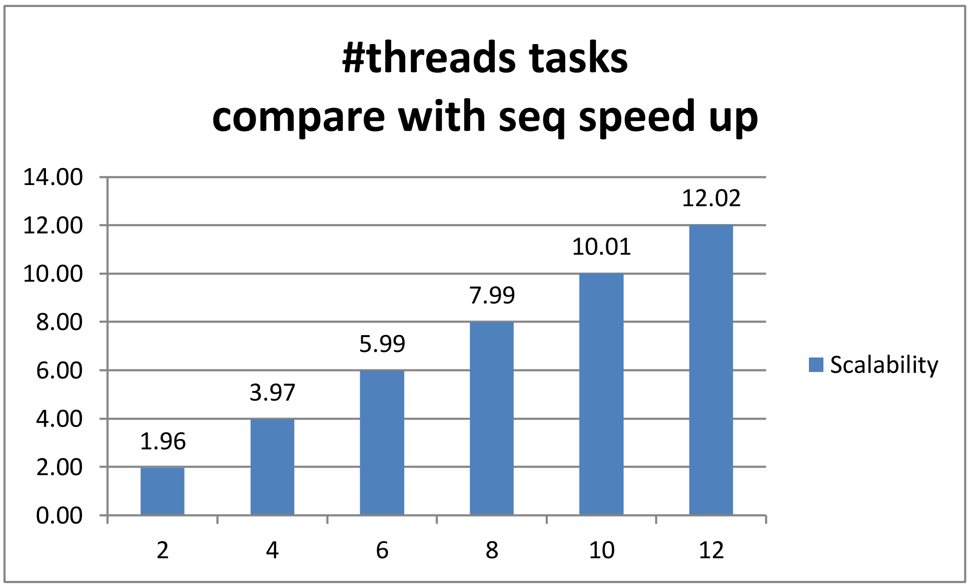
(Figure.3 **#process_#threads tasks compare with seq speed up** for hybird master do not compute pixels)



(Figure.4 **#threads tasks 數 load distribution (%)** for pthread)



(Figure.5 **#threads total time for seconds** for pthread)



(Figure.6 **#threads tasks compare with seq speed up** for pthread)

iii、Discussion(must base on the results in the plots)

(a). Compare and discuss the **scalability**of your implementations.

for hybird:

從 Figure2 及 Figure3 中可以看出來，越多 process(_前面)，速度大致上是越快。openmp 的 thread 也有發揮其讓計算更加平行的功能，有趣的事是不一定越多 threads 越好很可能是每個 process 要跟 master process 去 communication 越多 thread 跑去計算不一定是好事。

for pthread:

從 Figure5 及 Figure6 中可以看出來，越多 thread，越快且幾乎等

於理論最快的加速，也就是有多少 threads 就加速幾倍。**scalability 良好，可能是因為溝通成本**。真要細究可以發現越多 thread，越接近理論最快的加速，代表其時間分配恰當。

(b). Compare and discuss the**load balance**of your implementations.
for hybird:
從 Figure1 中可以看出來，越多 process(_前面)，分布的 load 越是不平衡。這是合理的，因為我的 worker 是做完事情再跟 master rank 再去要求新單，所以很容易如果一開始接到大單(需要較久的)，就會少接很多單，讓其他一開始接到小單，容易在接到小單，雖然接到要算久算短的單應該是要按照畫素的計算需求機率應該是固定的，但是期望值是機率乘上試驗次數，故一開始接到小單試驗次數就會變多，如果容易算的機率比來就比較大就很容易一開始就被搶單光。更甚者，還有 communication 成本，你用 mpi communication 少的話還要等其他人溝通完才能溝通，造成更嚴重的差距。
for pthread:
從 Figure4 中可以看出來，越多 thread，分布的 load 越是不平衡。道理基本上跟 hybird 版相似，但要考慮到自己創的 thread 是在 process 下主 thread 故其平衡性應該會更好，因為只在一個 node 上，communication 基本上沒成本。

iv、Others

- You are strongly encouraged to conduct more experiments and analysis of your implementations.

4. Experience & Conclusion

✓ Your conclusion of this assignment.
mpi 跟 openmp 一起用就是爽。好啦，pthread pool 其實也不錯用。主要是我覺得 pthread 如果用得恰當在單 node 上，可以很接近理論最快加速，mpi 跟 openmp 一起用需要考慮到溝通成本的影響，但是是不可或缺的，因為可以把計算資源盡可能的搾乾。

✓ What have you learned from this assignment?
pthread pool 的實現，如何混用 mpi openmp 在複雜的計算下。

✓ What difficulties did you encounter in this assignment?
當 pthread 到要與 mpi 跟 openmp 互動時，會有很多很奇怪的行為，就算你理論上把 pthread 的 work 限制在一個 process 還是一樣。
所以我還是無法把三者合在一起…，退而求其次，只做 master process 去控制其他 process。