Parallel Programming Homework 2 Report

Student ID: 105062635

Name: 吳浩寧

Design

1.) MS_MPI_static

將一張圖切割為許多個 grids，每個 grid 邊長為數個 pixels，以 round robin 法則分配工作。以下圖為例，一個 grid 邊長為 2 pixels，相同顏色代表分配到同一個 slave，由於每個 slave 分到的部分是固定的，因此可以獨立計算，不用靠 master 指派工作，只需在計算完後把結果傳回 master。

0,0	1,0	2,0	3,0	4,0
0,1	1,1	2,1	3,1	4,1
0,2	1,2	2,2	3,2	4,2
0,3	1,3	2,3	3,3	4,3
0,4	1,4	2,4	3,4	4,4

2.) MS_MPI_dynamic

以列做為分配工作的單位，第一輪 master 傳給每個 slave 一列 pixels，接著重複下列動作，master 進行自己的計算，詢問每個 slaves 計算狀況，傳送新的工作給計算完成的 slaves，直到每一列都計算完後，master 送出終止信號。

3.) MS_OpenMP_static

使用#pragma omp for collapse(2) schedule(static, 30)，將每 30 個 pixels 以 round robin 法則分配給每個 slave，計算完後將結果寫在共享的二維陣列中，最後一次將圖案畫出來。

4.) MS_OpenMP_dynamic

使用#pragma omp for collapse(2) schedule(guided, 10)，以 10 個 pixels 為單位動態分配工作，計算完的 threads 才會拿到新的工作，隨時間運行分配的單位則以指數遞減，計算完後將結果寫在共享的二維陣列中，最後一次將圖案畫出來。

5.) MS_Hybrid_static

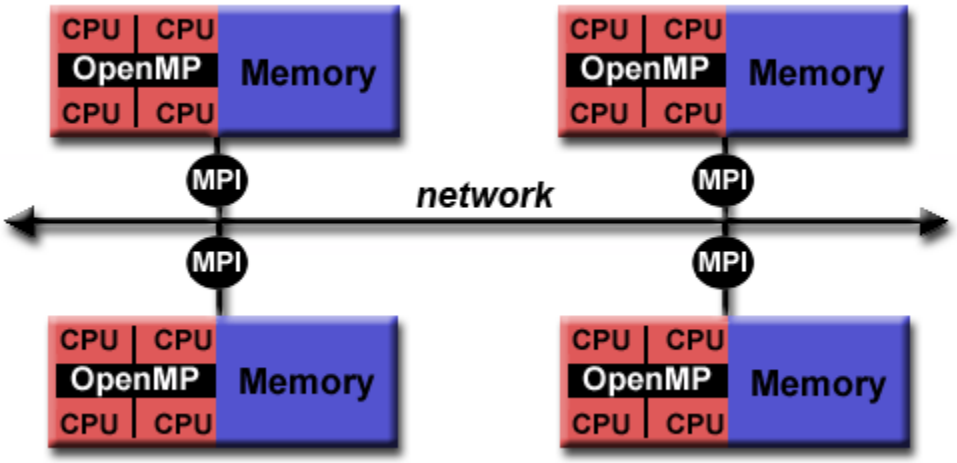
綜合 MPI、OpenMP 的 static 版本，差異之處只在每個 slave node 計算完會先存在自己獨立的記憶體中，最後利用 MPI_Reduce()將結果一列一列收集起來。

6.) MS_Hybrid_dynamic

綜合 MPI、OpenMP 的 dynamic 版本，我改進 master 瓶頸的辦法是讓 root node 做少一點事，此外我將圖形的上半部採用 static 的方式分配，以減少 communication 的次數，最後同樣利用 MPI_Reduce()將結果一列一列收集起來。

Performance analysis
待補><

Experience
這次實驗讓我更認識了 shared memory 和 message passing 的特性，MPI 就是 message passing 的代表；OpenMP 則是 shared memory，Hybrid 版本則是如下圖的 NUMA 架構。



從實驗結果發現，我的 static 版本跑得都比 dynamic 要快，首先，從產生的碎形可以觀察到，計算次數的分布是有 locality 的，因此依照我 static 的分配方式可以很均勻將工作平分，加上 static 版本每個 node，可以事先知道自己計算的部分，因此花在 communication 上的時間也少很多；相較之下，dynamic 版本需花大量時間在 communication 上，而且 master 的計算時間也會成為瓶頸，導致速度實際取決於 master 分到的工作量，若有辦法將等待結果，和進行計算兩部份平行在 master 執行，想必就能解決此問題，但我目前尚未想到如何實作。
這次作業遇到比較大的疑問，就是不知道有無方法可在 OpenMP 產生的 threads 中使用 MPI 的 API，在網路上本來有查到用 MPI_Init_thread 取代 MPI_Init 就能達成，但實際試了之後發現仍然不可行。