Problem descriptions

在這個 Assignment 裡，我們要用三種 MPI 提供的 API 去實作平行程式的運行，分別是 MPI_Send()+MPI_Recv()、MPI_Isend()+MPI_Irecv()、MPI_Scatter()+MPI_Gather()。而主要要求是從 Rank 0(Master)傳送資料到 Rank1~4(Slave)，他們進行接收資料後印出再傳出資料讓 Master 可以收到 Slave 傳輸的資料。所以一共需要 5 個 processer 去執行對應的 program，而 Master 傳出的資料為“Hi rank [slave_rank], I'm 廖洺玄 from Parallel Programming Design Course in 2024 “；Slave 傳出去的資料為“Rank [slave_rank] received. Thank you. “。在印出資料的同時也需要印出 MPI_Wtime()，來讓我們知道這個 program 在 master 或是 slave 的每一次收到資料花費多少時間。

Code and explanations

PPD_PA1_B0829002_1.c

```c
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int main(int argc, char** argv) {
    int proId = 0, numPro = 0;
    double startTime;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &proId);
    MPI_Comm_size(MPI_COMM_WORLD, &numPro);
    startTime = MPI_Wtime();
    char buffer[1024][numPro];
    if(proId == 0) {
        // this section is for node manager
        for(int i=1;i<numPro;i++) {
            // request part
            sprintf(buffer[i], "Hi rank %d, I'm 廖洺玄 from Parallel Programming Design Course in 2024", i);
            MPI_Send(buffer[i], strlen(buffer[i]), MPI_BYTE, i, 0, MPI_COMM_WORLD);
            // get response part
            char response[1024];
            MPI_Recv(response, 1024, MPI_BYTE, i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            printf("[MPI_Wtime():%f] Rank 0 got message from rank %d: '%s'\n", MPI_Wtime() - startTime, i, response);
        }
    } else {
        // this section is for execution of the node children
        // recieve part
        char data[1024] = {0};
        MPI_Recv(data, 1024, MPI_BYTE, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf("[MPI_Wtime():%f] Rank %d got message from rank 0: '%s'\n", MPI_Wtime() - startTime , proId, data);
        // response part
        sprintf(data, "Rank %d received. Thank you!", proId);
        MPI_Send(data, strlen(data), MPI_BYTE, 0, 0, MPI_COMM_WORLD);
    }
    // free the buffer
    MPI_Finalize();
}

```

Import library 和變數宣告

這部分用於印入 C 的 Standard library 和本次作業用到最重要的 MPI library，然後 Main function 會把使用者執行程式後面連帶的 arguments 引入讓 MPI 在 initialize parallel program 的時候可以使用到。proId、numPro 用來儲存下面再執行 parallel program 的時候是哪個 node 在執行、以及共有多少 node。double 的變數用於記錄 parallel program 開始時的時間。

```c
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int main(int argc, char** argv) {
    int proId = 0, numPro = 0;
    double startTime;
```

初始化 Parallel Program

這部分告訴 MPI 從哪裡開始執行平行的運算，然後給他 proId、numPro 用來記錄開出來的 node 的 process id 和共有多少個 node (process)需要初始化，並在一切初始化完成後開始執行 parallel program 並記錄下開始時間。

```c
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &proId);
MPI_Comm_size(MPI_COMM_WORLD, &numPro);
startTime = MPI_Wtime();
```

Master Parallel Program

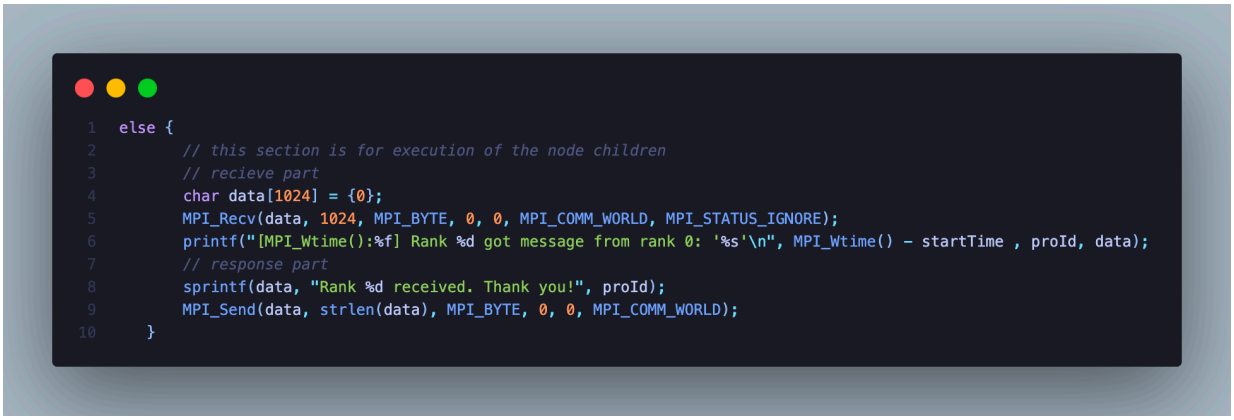
首先先給一個共用 numPro 個的 char array 讓他們可以儲存等一下要送出去的 data，sprintf 會把要傳給 slave 的資料印到 buffer 的 memory address 上，接下來就將資料傳給

第 i 個 slave，因為是 synchronous 的傳資料所以會等到 slave 接收到才往下等待接收那個 slave 傳過來的資料。收到後把 slave 傳過來的資料印出。

```
char buffer[1024][numPro];
if(proId == 0) {
    // this section is for node manager
    for(int i=1;i<numPro;i++) {
        // request part
        sprintf(buffer[i], "Hi rank %d, I'm 廖洺玄 from Parallel Programming Design Course in 2024", i);
        MPI_Send(buffer[i], strlen(buffer[i]), MPI_BYTE, i, 0, MPI_COMM_WORLD);
        // get response part
        char response[1024];
        MPI_Recv(response, 1024, MPI_BYTE, i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf("[MPI_Wtime():%f] Rank 0 got message from rank %d: '%s'\n", MPI_Wtime() - startTime, i, response);
    }
}
```

Slave Parallel Program

先 initialize 一個空的 data 讓 slave 可以接收從 master 傳過來的資料。MPI_Recv 收到資料後會讓 Master 的 parallel program 可以繼續，接下來 slave 會把花費時間和收到的資料印出。最後把要傳給 master 的資料印到 data's memory address 上並傳給 master 後等待，master 接收。



```
else {
        // this section is for execution of the node children
        // recieve part
        char data[1024] = {0};
        MPI_Recv(data, 1024, MPI_BYTE, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf("[MPI_Wtime():%f] Rank %d got message from rank 0: '%s'\n", MPI_Wtime() - startTime , proId, data);
        // response part
        sprintf(data, "Rank %d received. Thank you!", proId);
        MPI_Send(data, strlen(data), MPI_BYTE, 0, 0, MPI_COMM_WORLD);
    }
```

定義 Parallel Program 在哪部分結束

```
MPI_Finalize();
```

PPD_PA1_B0829002_2.c

```c
#include <stdio.h>
#include <string.h>
#include "mpi.h"

// Reference: https://zhuanlan.zhihu.com/p/362896326

int main(int argc, char** argv) {
    int proId = 0, numPro = 0;
    double startTime;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &proId);
    MPI_Comm_size(MPI_COMM_WORLD, &numPro);
    startTime = MPI_Wtime();
    char buffer[1024][numPro];
    if(proId == 0) {
        // this section is for node manager
        MPI_Request request[numPro-1];

        for(int i=1;i<numPro;i++) {
            // request part
            sprintf(buffer[i], "Hi rank %d, I'm 廖洺玄 from Parallel Programming Design Course in 2024", i);
            MPI_Isend(buffer[i], strlen(buffer[i]), MPI_BYTE, i, 0, MPI_COMM_WORLD, &request[i-1]);
            // get response part
            char response[1024];
            MPI_Irecv(response, 1024, MPI_BYTE, i, 0, MPI_COMM_WORLD, &request[i-1]);
            MPI_Wait(&request[i-1], MPI_STATUS_IGNORE);
            printf("[MPI_Wtime():%f] Rank 0 got message from rank %d: '%s'\n", MPI_Wtime() - startTime, i, response);
        }
        //MPI_Waitall(numPro-1, request, MPI_STATUS_IGNORE);
    } else {
        // this section is for execution of the node children
        // recieve part
        char data[1024] = {0};
        MPI_Request request;
        MPI_Irecv(data, 1024, MPI_BYTE, 0, 0, MPI_COMM_WORLD, &request);
        MPI_Wait(&request, MPI_STATUS_IGNORE);
        printf("[MPI_Wtime():%f] Rank %d got message from rank 0: '%s'\n", MPI_Wtime() - startTime, proId, data);
        // response part
        sprintf(data, "Rank %d received. Thank you!", proId);
        MPI_Isend(data, strlen(data), MPI_BYTE, 0, 0, MPI_COMM_WORLD, &request);
    }
    MPI_Finalize();
}
```

Import library 、變數宣告和初始化 Parallel Program

同 PPD_PA1_B0829002_1.c [按我到之前的解釋]

```c
#include <stdio.h>
#include <string.h>
#include "mpi.h"

// Reference: https://zhuanlan.zhihu.com/p/362896326

int main(int argc, char** argv) {
    int proId = 0, numPro = 0;
    double startTime;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &proId);
    MPI_Comm_size(MPI_COMM_WORLD, &numPro);
    startTime = MPI_Wtime();
```

Master Parallel Program

這邊和前面的方法不太一樣，因為要用到 Asynchronous 的通訊，所以我們要設一個 MPI_Request type 的變數去確認他已經收到 data 可以進行下一步了，而從 master 送資料到 slave 的流程大致上和先前的 program 差不多，唯一有差的地方是要帶&request[i-1] 讓之後收到的 slave 可以確認偵測/等待函數用的。之後便會開始接收從 slave 傳回來的資料，MPI_Wait 確認收到後 request 後便會把收到的 data 印出。

```
char buffer[1024][numPro];
if(proId == 0) {
    // this section is for node manager
    MPI_Request request[numPro-1];

    for(int i=1;i<numPro;i++) {
        // request part
        sprintf(buffer[i], "Hi rank %d, I'm 廖洺玄 from Parallel Programming Design Course in 2024", i);
        MPI_Isend(buffer[i], strlen(buffer[i]), MPI_BYTE, i, 0, MPI_COMM_WORLD, &request[i-1]);
        // get response part
        char response[1024];
        MPI_Irecv(response, 1024, MPI_BYTE, i, 0, MPI_COMM_WORLD, &request[i-1]);
        MPI_Wait(&request[i-1], MPI_STATUS_IGNORE);
        printf("[MPI_Wtime():%f] Rank 0 got message from rank %d: '%s'\n", MPI_Wtime() - startTime, i, response);
    }
    //MPI_Waitall(numPro-1, request, MPI_STATUS_IGNORE);
```

Slave Parallel Program

這裡也跟先前的部分很像但不太一樣的地方是我們需要收 data 時確認 request 是否完全收完後把要傳送給 master 的資料印在 data memory address 上，並傳送回 master。

```
else {
        // this section is for execution of the node children
        // recieve part
        char data[1024] = {0};
        MPI_Request request;
        MPI_Irecv(data, 1024, MPI_BYTE, 0, 0, MPI_COMM_WORLD, &request);
        MPI_Wait(&request, MPI_STATUS_IGNORE);
        printf("[MPI_Wtime():%f] Rank %d got message from rank 0: '%s'\n", MPI_Wtime() - startTime, proId, data);
        // response part
        sprintf(data, "Rank %d received. Thank you!", proId);
        MPI_Isend(data, strlen(data), MPI_BYTE, 0, 0, MPI_COMM_WORLD, &request);
    }
```

定義 Parallel Program 在哪部分結束

```
MPI_Finalize();
```

PPD_PA1_B0829002_3.c

Import library 、變數宣告和初始化 Parallel Program

同 PPD_PA1_B0829002_1.c [按我到之前的解釋]，多了一個 define Macro 的去定義每個要傳送的字串最長長度。

```c
#include <stdio.h>
#include <string.h>
#include "mpi.h"
#define maxMessageLength 1024

int main(int argc, char** argv) {
    int proId = 0, numPro = 0;
    double startTime;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &proId);
    MPI_Comm_size(MPI_COMM_WORLD, &numPro);
    startTime = MPI_Wtime();
```

Master Parallel Program

在 Master 裡面我們先宣告好要傳送資料的 string 和之後 slave 要接收 master 傳送資料的變數以及最後 master 要接收從 slave 傳回來的資料。宣告好後，用 sprintf 把要傳送的 string 印到 sendBuffer 的 memory address 上，之後用 MPI_Scatter 傳送到每個 node。（Scatter 的詳細運作請看 Discussion）

```c
char sendBuffer[numPro][maxMessageLength];
char recvBuffer[maxMessageLength], slaveBuffer[maxMessageLength];
char gatherBuffer[numPro][maxMessageLength];
if (proId == 0) {
    // Only the root initializes the send buffer
    for (int i = 0; i < numPro; i++)
        snprintf(sendBuffer[i], maxMessageLength, "Hi rank %d, I'm 廖洺玄 from Parallel Programming Design Course in 2024", i);
}

MPI_Scatter(sendBuffer, maxMessageLength, MPI_CHAR, recvBuffer, maxMessageLength, MPI_CHAR, 0, MPI_COMM_WORLD);
```

在從 Slave Gather 後我們會把傳回來的資料要出來。

```c
if (proId == 0) {
    for(int i=1;i<numPro;i++)
        printf("[MPI_Wtime():%f] Rank 0 got message from rank %d: '%s'\n", MPI_Wtime() - startTime, i, gatherBuffer[i]);
}
```

Slave Parallel Program

Slave 把收到的資料印出來後，用 Gather 發送資料回 Master，讓 Master 有可以印出的接收資料。（Gather 的詳細運作請看 Discussion）

```c
if (proId != 0)
    printf("[MPI_Wtime():%f] Rank %d got message from rank 0: '%s'\n", MPI_Wtime() - startTime ,proId, recvBuffer);
snprintf(slaveBuffer, maxMessageLength, "Rank %d received. Thank you!", proId);
MPI_Gather(slaveBuffer, maxMessageLength, MPI_CHAR, gatherBuffer, maxMessageLength, MPI_CHAR, 0, MPI_COMM_WORLD);
```

定義 Parallel Program 在哪部分結束

```c
MPI_Finalize();
```

Sampled outputs

PPD_PA1_B0829002_1.c

```
❯ mpirun -n 5 ./a.out
[MPI_Wtime():0.000022] Rank 0 got message from rank 1: 'Rank 1 received. Thank you!'
[MPI_Wtime():0.000035] Rank 0 got message from rank 2: 'Rank 2 received. Thank you!'
[MPI_Wtime():0.000011] Rank 1 got message from rank 0: 'Hi rank 1, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000046] Rank 0 got message from rank 3: 'Rank 3 received. Thank you!'
[MPI_Wtime():0.000064] Rank 0 got message from rank 4: 'Rank 4 received. Thank you!'
[MPI_Wtime():0.000016] Rank 2 got message from rank 0: 'Hi rank 2, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000029] Rank 3 got message from rank 0: 'Hi rank 3, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000035] Rank 4 got message from rank 0: 'Hi rank 4, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
❯ mpirun -n 5 ./a.out
[MPI_Wtime():0.000010] Rank 1 got message from rank 0: 'Hi rank 1, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000021] Rank 0 got message from rank 1: 'Rank 1 received. Thank you!'
[MPI_Wtime():0.000035] Rank 0 got message from rank 2: 'Rank 2 received. Thank you!'
[MPI_Wtime():0.000047] Rank 0 got message from rank 3: 'Rank 3 received. Thank you!'
[MPI_Wtime():0.000023] Rank 2 got message from rank 0: 'Hi rank 2, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000035] Rank 3 got message from rank 0: 'Hi rank 3, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000069] Rank 0 got message from rank 4: 'Rank 4 received. Thank you!'
[MPI_Wtime():0.000041] Rank 4 got message from rank 0: 'Hi rank 4, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
```

```
❯ mpirun -n 5 ./a.out
[MPI_Wtime():0.000836] Rank 0 got message from rank 1: 'Rank 1 received. Thank you!'
[MPI_Wtime():0.000820] Rank 1 got message from rank 0: 'Hi rank 1, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000851] Rank 0 got message from rank 2: 'Rank 2 received. Thank you!'
[MPI_Wtime():0.000841] Rank 2 got message from rank 0: 'Hi rank 2, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000857] Rank 3 got message from rank 0: 'Hi rank 3, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000863] Rank 0 got message from rank 3: 'Rank 3 received. Thank you!'
[MPI_Wtime():0.000863] Rank 4 got message from rank 0: 'Hi rank 4, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000875] Rank 0 got message from rank 4: 'Rank 4 received. Thank you!'
```

PPD_PA1_B0829002_2.c

```
❯ mpirun -n 5 ./b.out
[MPI_Wtime():0.000010] Rank 1 got message from rank 0: 'Hi rank 1, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000021] Rank 0 got message from rank 1: 'Rank 1 received. Thank you!'
[MPI_Wtime():0.000034] Rank 0 got message from rank 2: 'Rank 2 received. Thank you!'
[MPI_Wtime():0.000045] Rank 0 got message from rank 3: 'Rank 3 received. Thank you!'
[MPI_Wtime():0.000020] Rank 2 got message from rank 0: 'Hi rank 2, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000031] Rank 3 got message from rank 0: 'Hi rank 3, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000035] Rank 4 got message from rank 0: 'Hi rank 4, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000065] Rank 0 got message from rank 4: 'Rank 4 received. Thank you!'
❯ mpirun -n 5 ./b.out
[MPI_Wtime():0.000010] Rank 1 got message from rank 0: 'Hi rank 1, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000021] Rank 0 got message from rank 1: 'Rank 1 received. Thank you!'
[MPI_Wtime():0.000046] Rank 0 got message from rank 2: 'Rank 2 received. Thank you!'
[MPI_Wtime():0.000017] Rank 2 got message from rank 0: 'Hi rank 2, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000036] Rank 3 got message from rank 0: 'Hi rank 3, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000059] Rank 0 got message from rank 3: 'Rank 3 received. Thank you!'
[MPI_Wtime():0.000078] Rank 0 got message from rank 4: 'Rank 4 received. Thank you!'
[MPI_Wtime():0.000071] Rank 4 got message from rank 0: 'Hi rank 4, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
❯ mpirun -n 5 ./b.out
[MPI_Wtime():0.000021] Rank 0 got message from rank 1: 'Rank 1 received. Thank you!'
[MPI_Wtime():0.000035] Rank 0 got message from rank 2: 'Rank 2 received. Thank you!'
[MPI_Wtime():0.000009] Rank 1 got message from rank 0: 'Hi rank 1, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000056] Rank 0 got message from rank 3: 'Rank 3 received. Thank you!'
[MPI_Wtime():0.000067] Rank 0 got message from rank 4: 'Rank 4 received. Thank you!'
[MPI_Wtime():0.000021] Rank 2 got message from rank 0: 'Hi rank 2, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000021] Rank 3 got message from rank 0: 'Hi rank 3, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000059] Rank 4 got message from rank 0: 'Hi rank 4, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
```

PPD_PA1_B0829002_3.c

```
❯ mpirun -n 5 ./c.out
[MPI_Wtime():0.000013] Rank 1 got message from rank 0: 'Hi rank 1, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000014] Rank 3 got message from rank 0: 'Hi rank 3, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000048] Rank 0 got message from rank 1: 'Rank 1 received. Thank you!'
[MPI_Wtime():0.000051] Rank 0 got message from rank 2: 'Rank 2 received. Thank you!'
[MPI_Wtime():0.000052] Rank 0 got message from rank 3: 'Rank 3 received. Thank you!'
[MPI_Wtime():0.000053] Rank 0 got message from rank 4: 'Rank 4 received. Thank you!'
[MPI_Wtime():0.000027] Rank 2 got message from rank 0: 'Hi rank 2, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000029] Rank 4 got message from rank 0: 'Hi rank 4, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
❯ mpirun -n 5 ./c.out
[MPI_Wtime():0.000011] Rank 1 got message from rank 0: 'Hi rank 1, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000017] Rank 3 got message from rank 0: 'Hi rank 3, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000054] Rank 0 got message from rank 1: 'Rank 1 received. Thank you!'
[MPI_Wtime():0.000056] Rank 0 got message from rank 2: 'Rank 2 received. Thank you!'
[MPI_Wtime():0.000057] Rank 0 got message from rank 3: 'Rank 3 received. Thank you!'
[MPI_Wtime():0.000058] Rank 0 got message from rank 4: 'Rank 4 received. Thank you!'
[MPI_Wtime():0.000031] Rank 2 got message from rank 0: 'Hi rank 2, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000026] Rank 4 got message from rank 0: 'Hi rank 4, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
❯ mpirun -n 5 ./c.out
[MPI_Wtime():0.000012] Rank 1 got message from rank 0: 'Hi rank 1, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000015] Rank 3 got message from rank 0: 'Hi rank 3, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000045] Rank 0 got message from rank 1: 'Rank 1 received. Thank you!'
[MPI_Wtime():0.000048] Rank 0 got message from rank 2: 'Rank 2 received. Thank you!'
[MPI_Wtime():0.000049] Rank 0 got message from rank 3: 'Rank 3 received. Thank you!'
[MPI_Wtime():0.000049] Rank 0 got message from rank 4: 'Rank 4 received. Thank you!'
[MPI_Wtime():0.000029] Rank 2 got message from rank 0: 'Hi rank 2, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000028] Rank 4 got message from rank 0: 'Hi rank 4, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
❯ mpirun -n 5 ./c.out
[MPI_Wtime():0.000011] Rank 1 got message from rank 0: 'Hi rank 1, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000017] Rank 4 got message from rank 0: 'Hi rank 4, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000048] Rank 2 got message from rank 0: 'Hi rank 2, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000026] Rank 3 got message from rank 0: 'Hi rank 3, I'm 廖洺玄 from Parallel Programming Design Course in 2024'
[MPI_Wtime():0.000105] Rank 0 got message from rank 1: 'Rank 1 received. Thank you!'
[MPI_Wtime():0.000110] Rank 0 got message from rank 2: 'Rank 2 received. Thank you!'
[MPI_Wtime():0.000111] Rank 0 got message from rank 3: 'Rank 3 received. Thank you!'
[MPI_Wtime():0.000112] Rank 0 got message from rank 4: 'Rank 4 received. Thank you!'
```

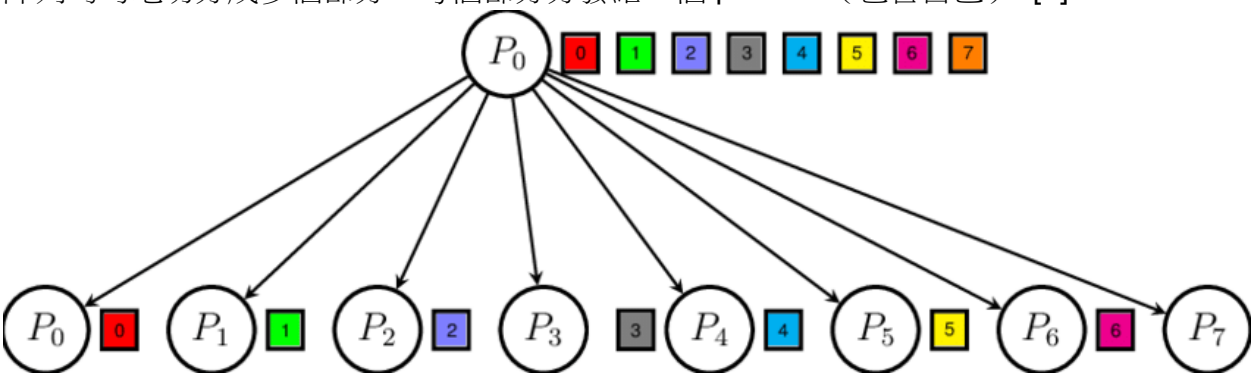
Discussions or what I've learned

在這個 Assignment 裡面我學習到了如何利用 MPI 提供的 API 開 muti-thread，讓我的 Program 可以平行運算。除此之外，之前在 Unix 修過的 thread 和 inter-process communication 的 Blocking 和 Non-blocking 也在第一個和第二個 program 用到了，雖然概念很像但是還是有一些值得注意的地方就是 thread.h 和 mpi.h 兩個提供的，例如 thread.h 要用到 OS mutual exclusion 的概念實作，但 MPI 有提供一個 request 可以去檢查是否已經傳送完成，相對 thread.h 上方便很多。

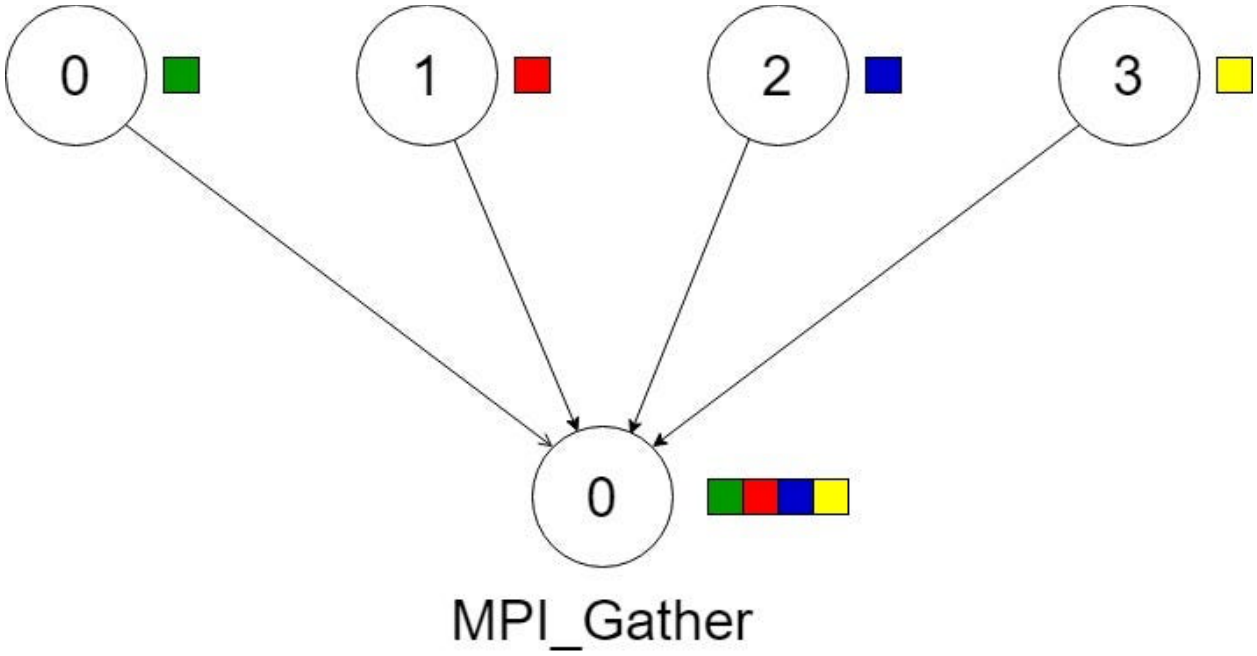
而在第一個和第二個 program 的 Blocking 和 Non-blocking，其中分別代表的通訊方式為：

1. Blocking（阻塞通訊）意味著調用的函數會一直等待，直到操作完成後才返回控制權給 program。這種方式在通訊過程中，發送方和接收方都需要同步操作。例如：MPI_Send、MPI_Recv，阻塞通訊的主要缺點是效率問題，特別是在延遲較高的情況下，它可能導致計算資源的浪費，因為處理器在等待完成通訊時可能什麼也做不了。
2. Non-blocking（非阻塞通訊）非阻塞通訊允許 program 在通訊操作尚未完成的情況下繼續執行，這有助於重疊計算與通訊，從而提升 program 的整體性能。例如：MPI_Isend、MPI_Irecv，非阻塞通訊需要額外的管理，因為你必須使用 MPI_Wait 或 MPI_Test 等函數來確認操作是否完成。這種方式適合於那些異步操作中的應用，特別是那些計算和通訊可以有效重疊的情況。[1]

在 MPI_Scatter 和 MPI_Gather 的功能應改是我覺得最實用的兩個平行 API 了，除了大幅提高我程式的精簡度以外，他用在資料傳送和收集的概念也非常有趣。MPI_Scatter 將資料從 Master（or also called root）分發到參與的所有 processes 中，他的用法就是 Master 擁有一個包含資料的 Array 或 vector（如果用 C++寫得話），並將這個陣列均勻地切分成多個部分，每個部分分發給一個 process（包含自己）。[2]



MPI_Gather 則會所有 process 中的資料收集到一個 process 中，用法大概就是每個 process（包括 Master）將其資料發送到 master，master 將接收到的資料組合成一個大陣列。[3]



Reference

[1] M. Kirby, “IntroMPI.ppt,” CS 6230: High-Performance Computing and Parallelization – Introduction to MPI, https://users.cs.utah.edu/~kirby/classes/cs6230/IntroMPI.pdf (accessed Apr. 23, 2024).

[2] K. Hasanov and A. Lastovetsky, “Hierarchical optimization of MPI reduce algorithms,” *Lecture Notes in Computer Science*, pp. 21–34, 2015. doi:10.1007/978-3-319-21909-7_3

[3] N. Joram, “Scatter and gather in MPI,” Medium, https://medium.com/nerd-for-tech/scatter-and-gather-in-mpi-e66b69366ee3 (accessed Apr. 23, 2024).