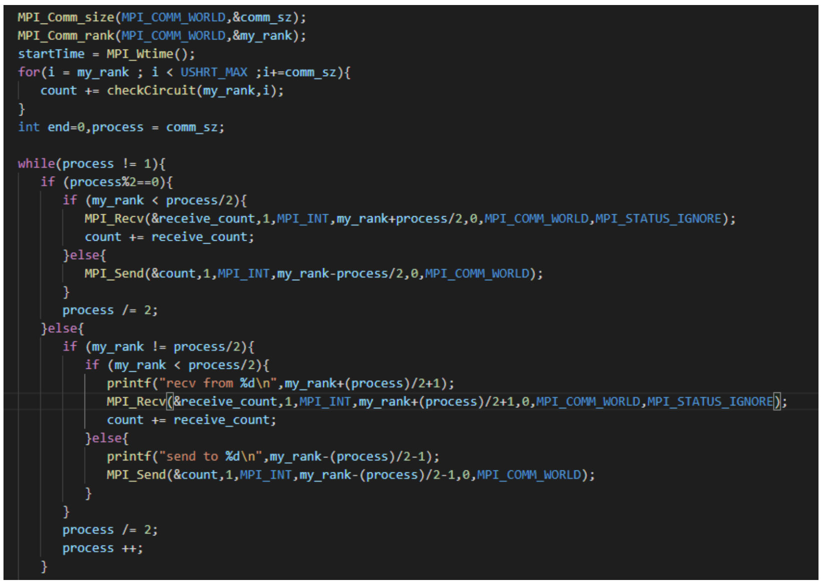
多處理機平行程式設計 2021 fall

作業一報告

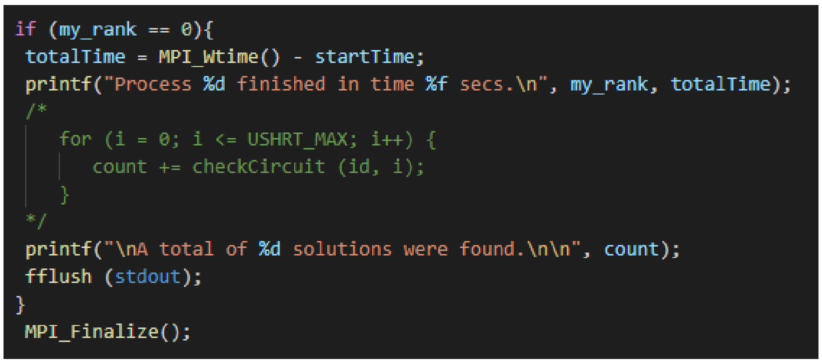
工科系劉彥甫 E94076178

第一題

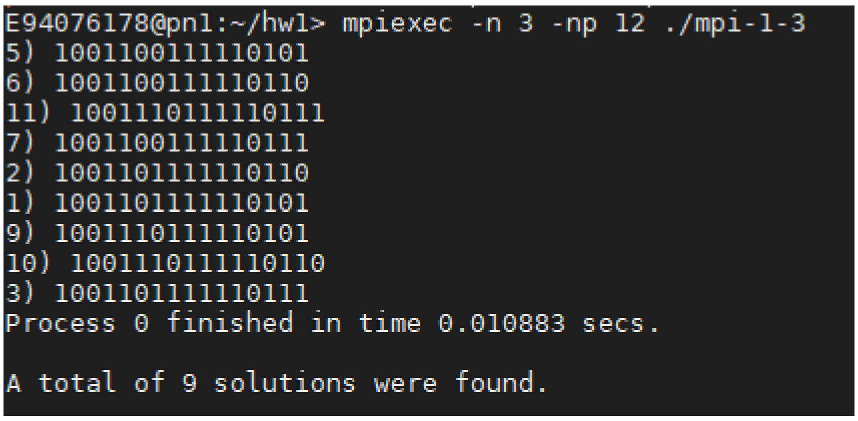
使用 alternative tree-structured 計算結果



```c
MPI_Comm_size(MPI_COMM_WORLD,&comm_sz);
MPI_Comm_rank(MPI_COMM_WORLD,&my_rank);
startTime = MPI_Wtime();
for(i = my_rank ; i < USHRT_MAX ;i+=comm_sz){
    count += checkCircuit(my_rank,i);
}
int end=0,process = comm_sz;

while(process != 1){
    if (process%2==0){
        if (my_rank < process/2){
            MPI_Recv(&receive_count,1,MPI_INT,my_rank+process/2,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
            count += receive_count;
        }else{
            MPI_Send(&count,1,MPI_INT,my_rank-process/2,0,MPI_COMM_WORLD);
        }
        process /= 2;
    }else{
        if (my_rank != process/2){
            if (my_rank < process/2){
                printf("recv from %d\n",my_rank+(process)/2+1);
                MPI_Recv(&receive_count,1,MPI_INT,my_rank+(process)/2+1,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
                count += receive_count;
            }else{
                printf("send to %d\n",my_rank-(process)/2-1);
                MPI_Send(&count,1,MPI_INT,my_rank-(process)/2-1,0,MPI_COMM_WORLD);
            }
        }
        process /= 2;
        process ++;
    }
```



```c
if (my_rank == 0){
 totalTime = MPI_Wtime() - startTime;
 printf("Process %d finished in time %f secs.\n", my_rank, totalTime);
 /*
    for (i = 0; i <= USHRT_MAX; i++) {
       count += checkCircuit (id, i);
    }
 */
 printf("\nA total of %d solutions were found.\n\n", count);
 fflush (stdout);
}
 MPI_Finalize();
```

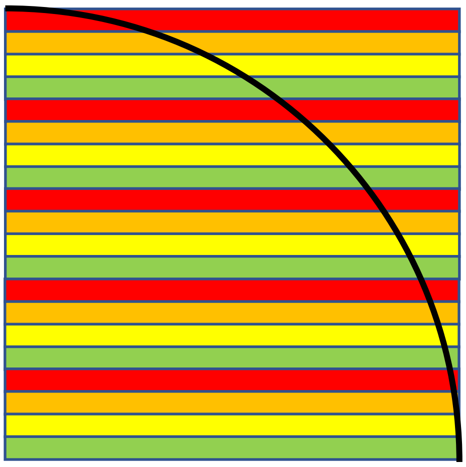


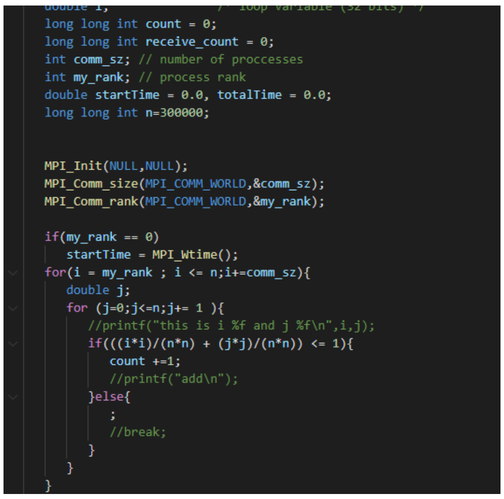
```
E94076178@pn1:~/hw1> mpiexec -n 3 -np 12 ./mpi-1-3
5) 1001100111110101
6) 1001100111110110
11) 1001110111110111
7) 1001100111110111
2) 1001101111110110
1) 1001101111110101
9) 1001110111110101
10) 1001110111110110
3) 1001101111110111
Process 0 finished in time 0.010883 secs.

A total of 9 solutions were found.
```

第二題

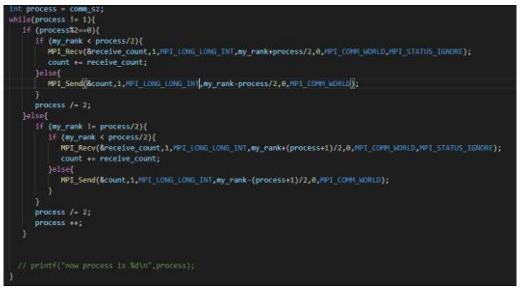
也是使用 alternative tree-structured 計算結果

資料切割方式(以 4 core 舉例)

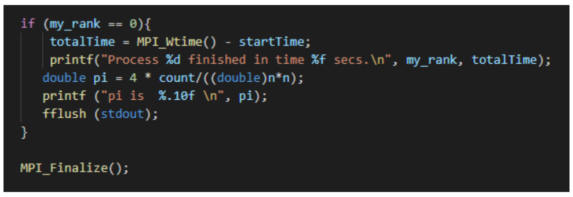




```
    double i;                /* loop variable (32 bits) */
    long long int count = 0;
    long long int receive_count = 0;
    int comm_sz; // number of proccesses
    int my_rank; // process rank
    double startTime = 0.0, totalTime = 0.0;
    long long int n=300000;


    MPI_Init(NULL,NULL);
    MPI_Comm_size(MPI_COMM_WORLD,&comm_sz);
    MPI_Comm_rank(MPI_COMM_WORLD,&my_rank);

    if(my_rank == 0)
        startTime = MPI_Wtime();
    for(i = my_rank ; i <= n;i+=comm_sz){
        double j;
        for (j=0;j<=n;j+= 1 ){
            //printf("this is i %f and j %f\n",i,j);
            if(((i*i)/(n*n) + (j*j)/(n*n)) <= 1){
                count +=1;
                //printf("add\n");
            }else{
                ;
                //break;
            }
        }
    }
```

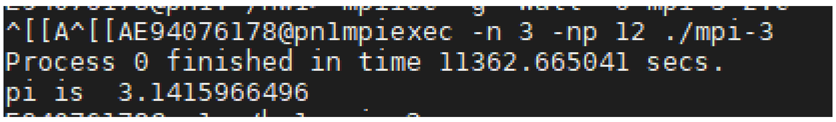


```
int process = comm_sz;
while(process != 1){
    if (process%2==0){
        if (my_rank < process/2){
            MPI_Recv(&receive_count,1,MPI_LONG_LONG_INT,my_rank+process/2,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
            count += receive_count;
        }else{
            MPI_Send(&count,1,MPI_LONG_LONG_INT,my_rank-process/2,0,MPI_COMM_WORLD);
        }
        process /= 2;
    }else{
        if (my_rank != process/2){
            if (my_rank < process/2){
                MPI_Recv(&receive_count,1,MPI_LONG_LONG_INT,my_rank+(process+1)/2,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
                count += receive_count;
            }else{
                MPI_Send(&count,1,MPI_LONG_LONG_INT,my_rank-(process+1)/2,0,MPI_COMM_WORLD);
            }
        }
        process /= 2;
        process ++;
    }


  // printf("now process is %d\n",process);
}
```



```
  if (my_rank == 0){
      totalTime = MPI_Wtime() - startTime;
      printf("Process %d finished in time %f secs.\n", my_rank, totalTime);
    double pi = 4 * count/((double)n*n);
    printf ("pi is  %.10f \n", pi);
    fflush (stdout);
  }

  MPI_Finalize();
```

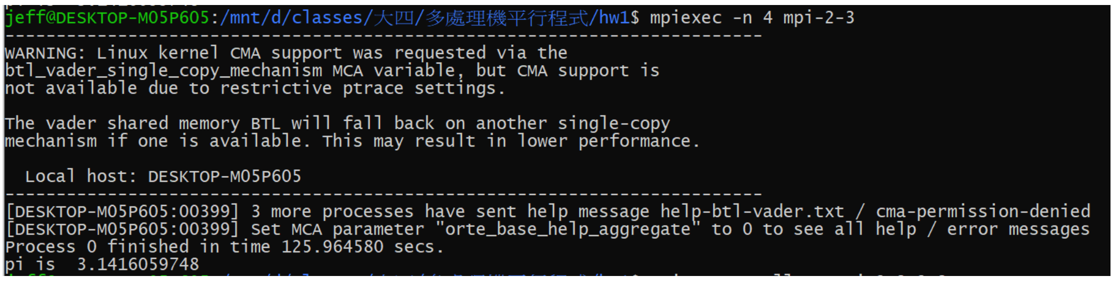
第二題結果(n=1000000)



```
^[[A^[[AE94076178@pnlmpiexec -n 3 -np 12 ./mpi-3
Process 0 finished in time 11362.665041 secs.
pi is  3.1415966496
```

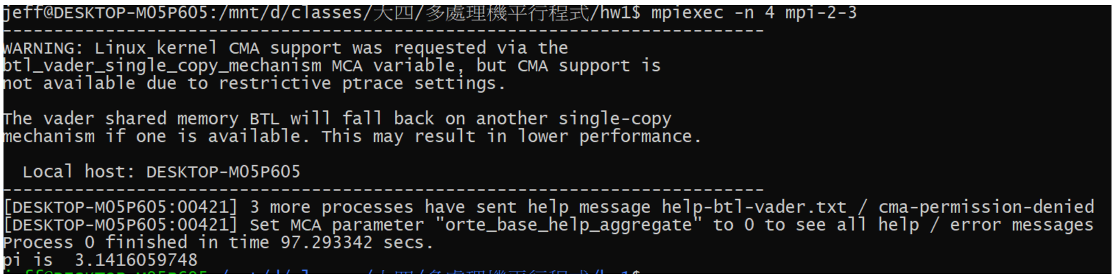
如果調整演算法，當測試點和原點距離>1 就馬上 break，可以縮短約 22%時間

原狀況



```
jeff@DESKTOP-M05P605:/mnt/d/classes/大四/多處理機平行程式/hw1$ mpiexec -n 4 mpi-2-3
--------------------------------------------------------------------------
WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

  Local host: DESKTOP-M05P605
--------------------------------------------------------------------------
[DESKTOP-M05P605:00399] 3 more processes have sent help message help-btl-vader.txt / cma-permission-denied
[DESKTOP-M05P605:00399] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
Process 0 finished in time 125.964580 secs.
pi is  3.1416059748
```

演算法調整後的狀況

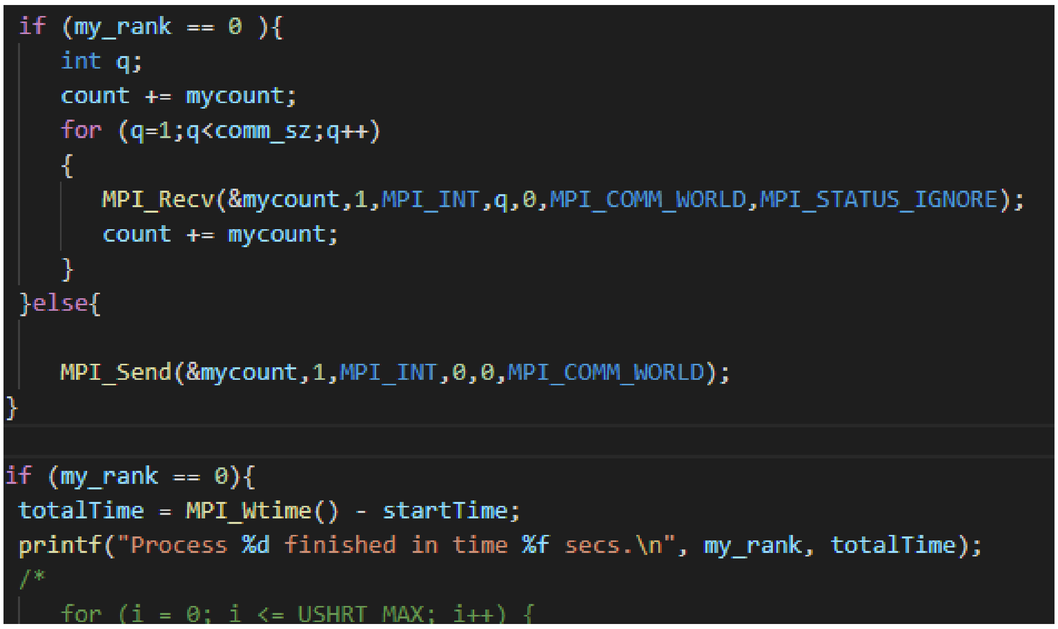


```
jeff@DESKTOP-M05P605:/mnt/d/classes/大四/多處理機平行程式/hw1$ mpiexec -n 4 mpi-2-3
--------------------------------------------------------------------------
WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

  Local host: DESKTOP-M05P605
--------------------------------------------------------------------------
[DESKTOP-M05P605:00421] 3 more processes have sent help message help-btl-vader.txt / cma-permission-denied
[DESKTOP-M05P605:00421] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
Process 0 finished in time 97.293342 secs.
pi is  3.1416059748
```

Something interesting

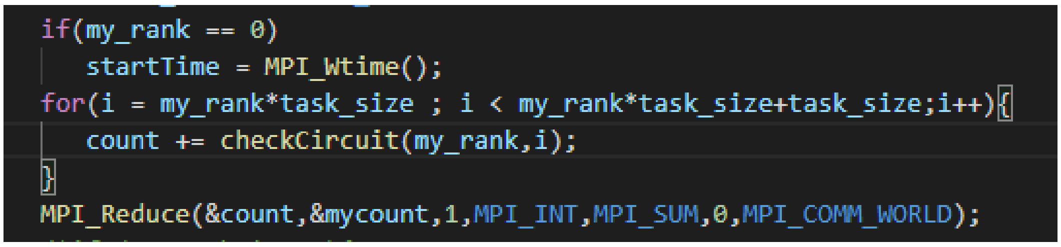
1. 第一題使用 mpi 反而會變慢，應該是運算需求不大反而花更多時間在資料傳送上
2. 兩題都有寫出 serial communication, reduce, tree structure 三種版本，在自己的電腦測試後(用 server 測試影響變數太多)差異不大，認為是資料交換需求不大，因此不會產生太大的影響

第一題 serial communication



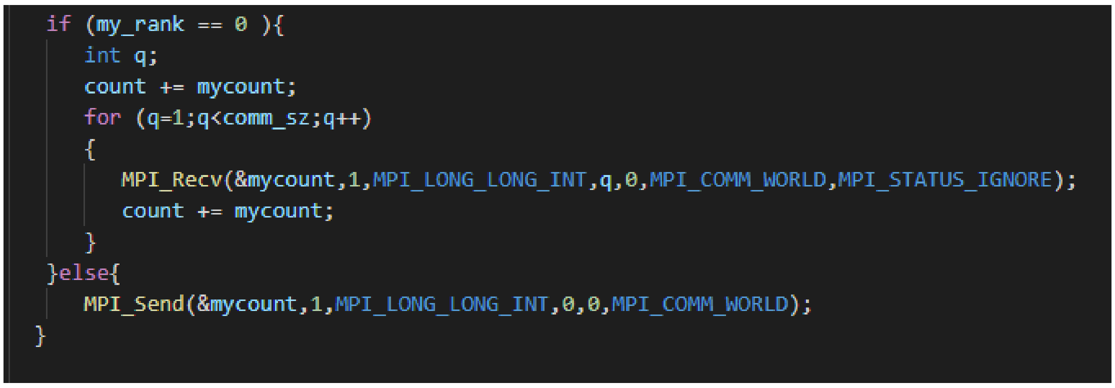
```c
 if (my_rank == 0 ){
    int q;
    count += mycount;
    for (q=1;q<comm_sz;q++)
    {
       MPI_Recv(&mycount,1,MPI_INT,q,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
       count += mycount;
    }
 }else{

    MPI_Send(&mycount,1,MPI_INT,0,0,MPI_COMM_WORLD);
}

if (my_rank == 0){
 totalTime = MPI_Wtime() - startTime;
 printf("Process %d finished in time %f secs.\n", my_rank, totalTime);
 /*
    for (i = 0; i <= USHRT_MAX; i++) {
```

reduce



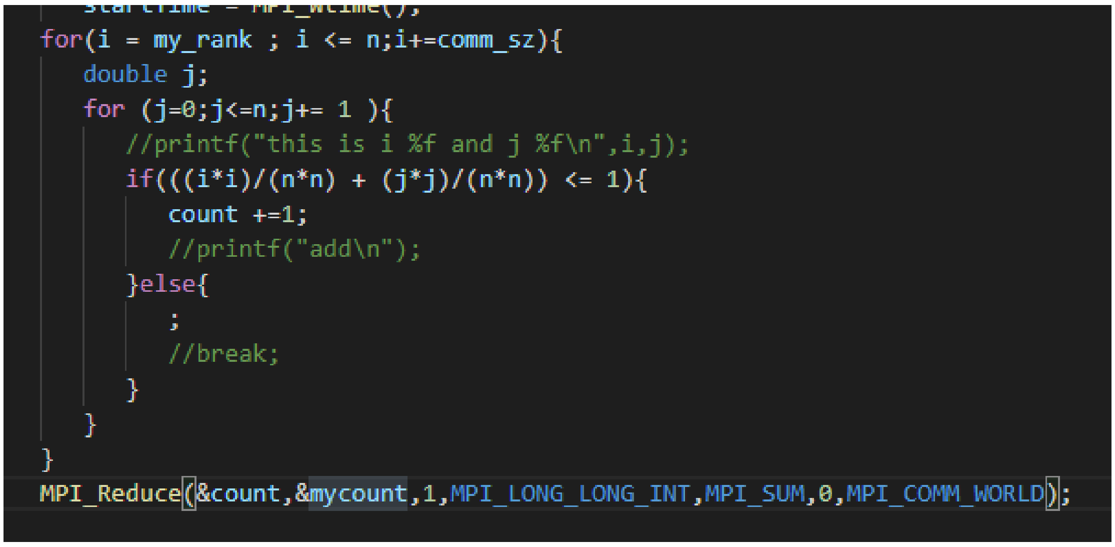
```
if(my_rank == 0)
    startTime = MPI_Wtime();
for(i = my_rank*task_size ; i < my_rank*task_size+task_size;i++){
    count += checkCircuit(my_rank,i);
}
MPI_Reduce(&count,&mycount,1,MPI_INT,MPI_SUM,0,MPI_COMM_WORLD);
```

第二題 serial communication



```
 if (my_rank == 0 ){
    int q;
    count += mycount;
    for (q=1;q<comm_sz;q++)
    {
        MPI_Recv(&mycount,1,MPI_LONG_LONG_INT,q,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
        count += mycount;
    }
 }else{
    MPI_Send(&mycount,1,MPI_LONG_LONG_INT,0,0,MPI_COMM_WORLD);
}
```

Reduce



```
    startTime = MPI_Wtime();
for(i = my_rank ; i <= n;i+=comm_sz){
    double j;
    for (j=0;j<=n;j+= 1 ){
        //printf("this is i %f and j %f\n",i,j);
        if(((i*i)/(n*n) + (j*j)/(n*n)) <= 1){
            count +=1;
            //printf("add\n");
        }else{
            ;
            //break;
        }
    }
}
MPI_Reduce(&count,&mycount,1,MPI_LONG_LONG_INT,MPI_SUM,0,MPI_COMM_WORLD);
```