

多處理機平行程式設計 HW1

資訊 113 I34096022 蔡易玟

Problem1

- What have you done?

首先，對多處理做基礎設定

```
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs); //total computer num
MPI_Comm_rank(MPI_COMM_WORLD,&id); //which computer

int each_comp_p = 65536/ numprocs;
```

接著，計算時間，分別在第一台電腦執行目標功能前、以及合併完所有電腦計算的數據之後

```
19 // time record
20 double time_start = 0.0, time_total = 0.0;
21 if(id==0){
22     time_start = MPI_Wtime();
23 }
```

```
47 // record time
48 if (id == 0) {
49     time_total = MPI_Wtime() - time_start;
50     //printf("pi is approximately %.16f, Error is %.16f\n",
51     //pi, fabs(pi - PI25DT));
52     printf("\nA total of %d solutions were found.\n\n", sum);
53     printf("total process time = %f\n", time_total);
54     fflush(stdout);
55 }
```

然後，執行每台電腦應該要執行的程式功能

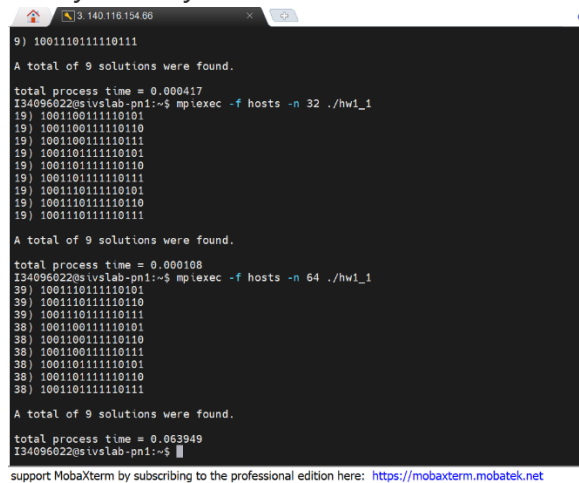
```
25 // which procs
26 for (i = each_comp_p*id; i <= each_comp_p*(id+1); i++) {
27     count += checkCircuit (id, i);
28 }
```

最後將每台電腦平行計算出的資料進行合併，合併方式類似於 mergesort

```
30 int merge_dest, merge2;
31 int num_i = numprocs, temp;
32 int sum=count;
33 while(1){
34     num_i = num_i/2;
35     if(id < num_i){
36         merge2 = id + num_i;
37         MPI_Recv(&temp, 1, MPI_INT, merge2, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
38         sum += temp;
39     } else if(id < 2*num_i){
40         merge_dest = id - num_i;
41         MPI_Send(&sum, 1, MPI_INT, merge_dest, 0, MPI_COMM_WORLD);
42     } else{
43         break;
44     }
45 }
```

最後執行，MPI_Finalize()，結束程式

- Analysis on your result?



```
9) 100110111110111
A total of 9 solutions were found.
total process time = 0.000417
I34096022@ivslab-pn1:~$ mpirun -f hosts -n 32 ./hw1_1
19) 1001100111110101
19) 1001100111110110
19) 1001100111110111
19) 1001101111110101
19) 1001101111110110
19) 1001101111110111
19) 1001110111110101
19) 1001110111110110
19) 1001110111110111
A total of 9 solutions were found.
total process time = 0.000108
I34096022@ivslab-pn1:~$ mpirun -f hosts -n 64 ./hw1_1
39) 1001110111110101
39) 1001110111110110
39) 1001110111110111
39) 1001100111110101
39) 1001100111110110
39) 1001100111110111
39) 1001101111110101
39) 1001101111110110
39) 1001101111110111
A total of 9 solutions were found.
total process time = 0.063949
I34096022@ivslab-pn1:~$
```

由指令可以看到，交由 32 台電腦進行平行處理時，都是由 Id 為 9 的電腦找到解法；而由 64 台電腦進行處理時分別由 Id 為 38, 39 的電腦找到解法。另外，可以看出找到的解法數量是相同的，但是由 32 台電腦平行處理所需的時間是少於 64 台電腦平行處理的。

- Any difficulties?

一開始不太懂如何使用多台電腦的平行處理指令，算是入門一個程式語言技術的門檻吧，這時候算是最痛苦，要了解語言的使用，更重要是他實現的邏輯是什麼。當然，搞懂之後就有越來越上手的趨勢。

- (optional) Feedback to Tas?

Problem2

- What have you done?

基本設定和架構與第一題幾乎一樣

```
17 MPI_Init(&argc,&argv);
18 MPI_Comm_size(MPI_COMM_WORLD,&numprocs); //total computer num
19 MPI_Comm_rank(MPI_COMM_WORLD,&id); //which computer
```

包含時間設定計量

```
32 // time record
33 double time_start = 0.0, time_total = 0.0;
34 if(id==0){
35     time_start = MPI_Wtime();
36 }
```

```
75 // record time
76 if (id == 0) {
77     time_total = MPI_Wtime() - time_start;
78     //printf("pi is approximately %.16f, Error is %.16f\n",
79     //pi, fabs(pi - PI25DT));
80     printf("\n Pi = %f \n\n", pi_estimate);
81     printf("total process time = %f\n", time_total);
82     fflush(stdout);
83 }
```

不一樣之處，在這一題我們需要輸入總共需要執行程式的次數，並將這個訊息通知給所有的平行處理電腦知道

```
21 if(id==0){
22     scanf("%lld", &num);
23     for(int i=1; i<numprocs; i++){
24         MPI_Send(&num,1,MPI_LONG_LONG, i ,0,MPI_COMM_WORLD);
25     }
26 } else{
27     MPI_Recv(&num, 1, MPI_LONG_LONG, 0, 0, MPI_COMM_WORLD,MPI_STATUS_IGNORE);
28 }
29
30 int each_comp_p = num/ numprocs;
```

本題執行的程式功能如下

```
39 // which procs
40 srand(time(NULL));
41 int number_in_circle = 0;
42 for (int toss = 0; toss < each_comp_p; toss++){
43     double x = rand();
44     double y = rand();
45     x = (x/RAND_MAX) *2 -1;
46     y = (y/RAND_MAX) *2 -1;
47     double distance_squared = x*x + y*y;
48     if (distance_squared <= 1) {
49         number_in_circle++;
50     }
51 }
52
```

最後亦須要將所有電腦平行劑量的數字做統整計算

```
54     int merge_dest, merge2;
55     int num_i = numprocs, temp;
56     int sum=number_in_circle; //
57     while(1){
58         num_i = num_i/2;
59         if(id < num_i){
60             merge2 = id + num_i;
61             MPI_Recv(&temp, 1, MPI_LONG_LONG, merge2, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
62             sum += temp;
63         } else if(id < 2*num_i){
64             merge_dest = id - num_i;
65             MPI_Send(&sum, 1, MPI_LONG_LONG, merge_dest, 0, MPI_COMM_WORLD);
66         } else{
67             break;
68         }
69     }
```

- Analysis on your result?



```
YOUR APPLICATION TERMINATED WITH THE EXIT STRING: Killed (signal 9)
This typically refers to a problem with your application.
Please see the FAQ page for debugging suggestions
I34096022@svslab-pn1:~$ mpirun -f hosts -n 32 ./hw1_2
100
    Pi = 3.840000
total process time = 0.058737
I34096022@svslab-pn1:~$ mpirun -f hosts -n 32 ./hw1_2
10000
    Pi = 3.120000
total process time = 0.044267
I34096022@svslab-pn1:~$ mpirun -f hosts -n 32 ./hw1_2
100000
    Pi = 3.114560
total process time = 0.130637
I34096022@svslab-pn1:~$ mpirun -f hosts -n 32 ./hw1_2
1000000
    Pi = 3.140384
total process time = 0.064676
I34096022@svslab-pn1:~$ mpirun -f hosts -n 32 ./hw1_2
10000000
    Pi = 3.141266
total process time = 0.158529
I34096022@svslab-pn1:~$
```

support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

由截圖可以看到當總執行次數太少(eg. 100)時，結果大於理想值，當總次數提升時目標結果數值遞減到低於理想值(3.14)，直到總次數到1000000 結果數值才真正趨近理想值。至於執行的時間可以看出，由執行次數少到多的過程，所花時間幾乎由峰值到谷底，再到峰值。

- Any difficulties?
有點久沒有寫 C 語言了，要特別注意 `type` 的指定，以免數值不準確。
- (optional) Feedback to Tas?