**高性能并行计算第6次作业**

姓名：代宏刚 学号：2020317110061

**代码地址：**

1.Greeting程序

1. #include <stdio.h>
2. #include <string.h>
3. #include "mpi.h"
5. **int** main(**int** argc,**char**\* argv[]) {
6. **int** numprocs, myid, source;
7. MPI\_Status status;
8. **char** message[100];
9. MPI\_Init(&argc, &argv);
10. MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myid);
11. MPI\_Comm\_size(MPI\_COMM\_WORLD, &numprocs);
12. **if**(myid!=0) {
13. sprintf(message,"greeting from procs %d\n",myid);
14. MPI\_Send(message,strlen(message)+1,MPI\_CHAR,0,99,MPI\_COMM\_WORLD);
15. } **else** {
16. **for**(source = 1; source<numprocs; source++) {
17. MPI\_Recv(message,100,MPI\_CHAR,source,99,MPI\_COMM\_WORLD,&status);
18. printf("%s\n",message);
19. }
20. }
21. MPI\_Finalize();
22. **return** 0;
23. }

2.点对点通信 pingpong

1. #include "mpi.h"
2. #include <stdio.h>
3. #include <stdlib.h>
4. **int** main(**int** argc, **char**\* argv[]) {
5. **const** **int** PING\_PONG\_LIMIT = 10;
6. MPI\_Init(&argc, &argv);
7. **int** world\_rank;
8. MPI\_Comm\_rank(MPI\_COMM\_WORLD, &world\_rank);
9. **int** world\_size;
10. MPI\_Comm\_size(MPI\_COMM\_WORLD, &world\_size);
11. **if**(world\_size!=2) {
12. fprintf(stderr, "World Size must be two for %s\n",argv[0]);
13. MPI\_Abort(MPI\_COMM\_WORLD, 1);
14. }
15. **int** ping\_pong\_count = 0;
16. **int** partner\_rank = (world\_rank+1)%2;
17. **while**(ping\_pong\_count<PING\_PONG\_LIMIT) {
18. **if**(world\_rank == ping\_pong\_count%2) {
19. ping\_pong\_count++;
20. MPI\_Send(&ping\_pong\_count,1,MPI\_INT,partner\_rank,0,MPI\_COMM\_WORLD);
21. printf("%d send and incremented ping\_pong\_count %d to %d\n",world\_rank,ping\_pong\_count,partner\_rank);
22. } **else** {
24. MPI\_Recv(&ping\_pong\_count,1,MPI\_INT,partner\_rank,0,MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);
25. printf("%d received ping\_pong\_count %d from %d\n", world\_rank, ping\_pong\_count, partner\_rank);
26. }
27. }
28. MPI\_Finalize();
29. **return** 0;
30. }

3.点对点通信结合与分解计算pi

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include "mpi.h"
4. **long** **long** num\_steps = 1000000000;
5. **double** step;
6. **int** main(**int** argc, **char**\* argv[]) {
7. **double** buff[1];
8. **double** t1, t2;
9. MPI\_Init(&argc, &argv);
10. t1 = MPI\_Wtime();
11. **int** world\_rank, world\_size;
12. MPI\_Comm\_rank(MPI\_COMM\_WORLD, &world\_rank);
13. MPI\_Comm\_size(MPI\_COMM\_WORLD, &world\_size);
14. **int** i;
15. **double** x,pi = 0.0;
16. step = 1.0/(**double**)num\_steps;
17. **for**(i=world\_rank; i<=num\_steps; i+=world\_size) {
18. x = (i-0.5)\*step;
19. buff[0] += 4.0/(1.0+x\*x);
20. }
21. buff[0] \*= step;
22. **if**(world\_rank!=0) {
23. MPI\_Send(buff,1,MPI\_DOUBLE,0,99,MPI\_COMM\_WORLD);
24. } **else** {
25. pi = buff[0];
26. **for**(i=1; i<world\_size; i++) {
27. MPI\_Recv(buff,1,MPI\_DOUBLE,MPI\_ANY\_SOURCE,99,MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);
28. pi += buff[0];
29. }
30. t2 = MPI\_Wtime();
31. printf("%d\t%f\t%f\n",world\_size,pi,t2-t1);
32. }
33. **return** MPI\_Finalize();
34. }

**实验结果：**

1.Greeting 程序

[2020317110061@bxjs hw06]$ mpirun -np 5 greeting1  
greeting from procs 1  
  
greeting from procs 2  
  
greeting from procs 3  
  
greeting from procs 4

使用mpirun 执行程序，设置进程数np 为 5, 1-4号进程向0号进程发送消息，0号进程使用MPI\_RECV方法按进程号接受消息。

2. 点对点通信 pingpong

[2020317110061@bxjs hw06]$ mpirun -np 2 pingpong  
0 send and incremented ping\_pong\_count 1 to 1  
0 received ping\_pong\_count 2 from 1  
0 send and incremented ping\_pong\_count 3 to 1  
0 received ping\_pong\_count 4 from 1  
0 send and incremented ping\_pong\_count 5 to 1  
0 received ping\_pong\_count 6 from 1  
0 send and incremented ping\_pong\_count 7 to 1  
0 received ping\_pong\_count 8 from 1  
0 send and incremented ping\_pong\_count 9 to 1  
0 received ping\_pong\_count 10 from 1  
1 received ping\_pong\_count 1 from 0  
1 send and incremented ping\_pong\_count 2 to 0  
1 received ping\_pong\_count 3 from 0  
1 send and incremented ping\_pong\_count 4 to 0  
1 received ping\_pong\_count 5 from 0  
1 send and incremented ping\_pong\_count 6 to 0  
1 received ping\_pong\_count 7 from 0  
1 send and incremented ping\_pong\_count 8 to 0  
1 received ping\_pong\_count 9 from 0  
1 send and incremented ping\_pong\_count 10 to 0

3.pi计算程序

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| np | pi | time(s) | speed\_ratio | efficiency |
| 1 | 3.141593 | 16.770370 | 1 | 100% |
| 2 | 3.141593 | 8.389627 | 1.99894 | 99.947% |
| 4 | 3.141593 | 4.196640 | 3.99614 | 99.9035% |
| 8 | 3.141593 | 2.099380 | 7.98825 | 99.853125% |
| 10 | 3.141593 | 1.682746 | 9.96607 | 99.6607% |
| 20 | 3.141593 | 0.856682 | 19.576 | 97.88% |
| 40 | 3.141593 | 0.443238 | 37.836 | 94.59% |

np:进程数

pi:计算得到的π值

time(s):程序执行的时间，单位为s

speed\_retio:加速比

efficiency：并行效率