**高性能并行计算第7次作业**

姓名：代宏刚 学号：2020317110061

**代码地址：**

1.MPI\_Bsend

1. #include <stdio.h>
2. #include "mpi.h"
3. #include <stdlib.h>
4. **long** **long** num\_steps = 1000000000;
5. **double** step;
6. **int** main(**int** argc, **char**\* argv[]) {
7. **double** \*sum;
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. sum = (**double**\*)malloc(**sizeof**(**double**)\*world\_size);
15. **int** i;
16. **double** x,pi = 0.0;
17. step = 1.0/(**double**)num\_steps;
18. **for**(i=world\_rank,sum[world\_rank]=0; i<=num\_steps; i+=world\_size) {
19. x = (i-0.5)\*step;
20. sum[world\_rank] += 4.0/(1.0+x\*x);
21. }
22. **for**(i=0; i<world\_size; i++)
23. MPI\_Bcast(sum+i,1,MPI\_DOUBLE,i,MPI\_COMM\_WORLD);
24. MPI\_Barrier(MPI\_COMM\_WORLD);
25. **if**(world\_rank==0) {
26. **for**(i=0; i<world\_size; i++) {
27. pi += sum[i]\*step;
28. }
29. t2 = MPI\_Wtime();
30. printf("%d\t%f\t%f\n",world\_size, pi,t2-t1);
31. }
32. **return** MPI\_Finalize();
33. }

2.MPI\_Gather

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

3.MPI\_Reduce

1. #include <stdio.h>
2. #include "mpi.h"
3. #include <stdlib.h>
4. **long** **long** num\_steps = 1000000000;
5. **double** step;
6. **int** main(**int** argc, **char**\* argv[]) {
7. **double** buff = 0.0;
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 += 4.0/(1.0+x\*x);
20. }
21. MPI\_Reduce(&buff,&pi,1,MPI\_DOUBLE,MPI\_SUM,0,MPI\_COMM\_WORLD);
22. MPI\_Barrier(MPI\_COMM\_WORLD);
23. **if**(world\_rank==0) {
24. pi \*= step;
25. t2 = MPI\_Wtime();
26. printf("%d\t%f\t%f\n",world\_size,pi,t2-t1);
27. }
28. **return** MPI\_Finalize();
29. }

**实验结果：**

①.MPI\_Bsend

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **np** | **pi** | **time** | **speed\_ratio** | **efficiency** |
| 1 | 3.141593 | 16.752599 | 1 | 100.0000% |
| 2 | 3.141593 | 8.3849 | 1.99795 | 99.8975% |
| 4 | 3.141593 | 4.25549 | 3.9367 | 98.4175% |
| 8 | 3.141593 | 2.128564 | 7.87038 | 98.3798% |
| 10 | 3.141593 | 1.690007 | 9.91274 | 99.1274% |
| 20 | 3.141593 | 0.862754 | 19.4176 | 97.0880% |
| 40 | 3.141593 | 0.453282 | 36.9584 | 92.3960% |

②.MPI\_Gather

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **np** | **pi** | **time** | **speed\_ratio** | **efficiency** |
| 1 | 3.141593 | 16.754297 | 1 | 100.0000% |
| 2 | 3.141593 | 8.386676 | 1.99773 | 99.8865% |
| 4 | 3.141593 | 4.201636 | 3.98757 | 99.6893% |
| 8 | 3.141593 | 2.100748 | 7.9754 | 99.6925% |
| 10 | 3.141593 | 1.679918 | 9.97328 | 99.7328% |
| 20 | 3.141593 | 0.856702 | 19.5567 | 97.7835% |
| 40 | 3.141593 | 0.443784 | 37.7533 | 94.3833% |

③.MPI\_Reduce

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **np** | **pi** | **time** | **speed\_ratio** | **efficiency** |
| 1 | 3.141593 | 16.767104 | 1 | 100.0000% |
| 2 | 3.141593 | 8.391957 | 1.998 | 99.9000% |
| 4 | 3.141593 | 4.196736 | 3.99527 | 99.8818% |
| 8 | 3.141593 | 2.102882 | 7.97339 | 99.6674% |
| 10 | 3.141593 | 1.688764 | 9.92862 | 99.2862% |
| 20 | 3.141593 | 0.855376 | 19.602 | 98.0100% |
| 40 | 3.141593 | 0.43097 | 38.9055 | 97.2638% |

np:进程数

pi:程序计算的pi值

time:程序执行时间

speed\_ratio:加速比

efficiency:并行效率