实验三

MPI 群集通信测试

实验指导书

1. 群集同步：**int MPI\_Barrier(MPI\_Comm comm)**

当**COMM**通信器中的所有进程都执行这个函数时才返回

if (rank==0)

{

double starttime = MPI\_Wtime();

//Sleep(1000);

MPI\_Barrier(MPI\_COMM\_WORLD);

double endtime=MPI\_Wtime();

cout<<rank<<" :time is : "<<endtime-starttime<<endl;

}

其他的进程各自休眠长短不同的时间。

各进程将等待最晚完成的进程一起返回。

1. 群集通信
   1. 广播

int rank, size, tag=1,source=0;

int n=666;

MPI\_Init(&*argc*, &*argv*); /\*MPI的初始化函数\*/

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); /\*该进程的编号\*/

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size); /\*总的进程数目\*/

if(rank==source)

{

cout<< "input n pls:"<<endl;

cin>>n; // 从根进程中获得数据。

}

MPI\_Bcast(&n, 1, MPI\_INT, 0, MPI\_COMM\_WORLD); // 注意所有的进程都要调用此函数。也相当于一个Barrier

if (rank==source)

{

double starttime = MPI\_Wtime();

//Sleep(1000);

cout<<rank<< " recv "<<n<<endl;

double endtime=MPI\_Wtime();

cout<<rank<<" :time is : "<<endtime-starttime<<endl;

}

if (rank==1)

{

double starttime = MPI\_Wtime();

Sleep(1000);

// 无需调用什么接收函数。

cout<<rank<< " recv "<<n<<endl;

double endtime= MPI\_Wtime();

cout<<rank<<" :time is : "<<endtime-starttime<<endl;

}

if (rank==2)

{

double starttime = MPI\_Wtime();

Sleep(2000);

cout<<rank<< " recv "<<n<<endl;

double endtime= MPI\_Wtime();

cout<<rank<<" :time is : "<<endtime-starttime<<endl;

}

MPI\_Finalize();

return 0;

* 1. 散发 **MPI\_Scatter**

int rank, size, i,j;

int table[4][4];

int row[4];

MPI\_Init( &*argc*, &*argv* );

MPI\_Comm\_rank( MPI\_COMM\_WORLD, &rank );

MPI\_Comm\_size( MPI\_COMM\_WORLD, &size );

int send\_count = 4;

int recv\_count = 4;

if (rank == 0)

for ( i=0; i<4; i++)

for ( j=0; j<4; j++ )

table[i][j] = i+j;

MPI\_Scatter(&table[0][0], send\_count, MPI\_INT,&row[0] , recv\_count, MPI\_INT, 0, MPI\_COMM\_WORLD);

if(rank==1) // 0,1,2,3 分别测试。

for(int i=0;i<4;i++)

cout<<row[i]<<endl;

MPI\_Finalize();

return 0;

程序把一个表格

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |
| 1 | 2 | 3 | 4 |
| 2 | 3 | 4 | 5 |
| 3 | 4 | 5 | 6 |

第i行分给第i个进程。

* 1. 收集

MPI\_Gather 过程与以上相反，不再举例。