作业六 进程环

150******* 李师尧

改写 PDF 文档 4.3.1 中的示例程序。从 mpiexec 创建的进程组中，使用 MPI_Comm_split()实现示例程序要求的“环”。若 mpiexec 所创建的进程数不为 3 的整数倍，则将序号高的进程余留出来不参加“环”。

首先根据每个进程的 myid 将大于 3n 的进程排除掉，然后利用 mpi_comm_split 函数将进程分组，然后利用 mpi_intercomm_create()和 mpi_intercomm_merge()创建足间和组内通信子，最后利用 MPI_Cart_create()得到笛卡尔坐标。关键代码段，如图 1 所示，其余代码见附页。

另外，老师给的 pdf 的例子中，需要用 mpi_any_source 而不是用 i，否则会堵塞。

```
MPI_Comm_split( MPI_COMM_WORLD, keyid, myid, &intra_gcomm );
if ( keyid == 0 )
{
    MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 1, 1, &inter_lcomm );
    MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 2, 2, &inter_rcomm );
    MPI_Intercomm_merge( inter_lcomm, keyid, &intra_lcomm );
    MPI_Intercomm_merge( inter_rcomm, keyid, &intra_rcomm );
    printf("keyid -%d\n",myid);
}
else if ( keyid == 1 )
{
    MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 0, 1, &inter_rcomm );
    MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 2, 3, &inter_lcomm );
    MPI_Intercomm_merge( inter_rcomm, keyid, &intra_rcomm );
    MPI_Intercomm_merge( inter_lcomm, keyid, &intra_lcomm );
    printf("keyid -%d\n",myid);
}
else if (keyid == 2 )
{
    MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 0, 3, &inter_lcomm );
    MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 1, 3, &inter_rcomm );
    MPI_Intercomm_merge( inter_lcomm, keyid, &intra_lcomm );
    MPI_Intercomm_merge( inter_rcomm, keyid, &intra_rcomm );
    printf("keyid -%d\n",myid);
}
```

图 1 关键代码段

```
#include<stdio.h>
#include<string.h>
#include<mpi.h>

int main(int argc, char ** argv)
{
    MPI_Comm intra_gcomm, inter_lcomm, inter_rcomm, intra_lcomm, intra_rcomm;
    MPI_Status status;
    int keyid, flag;
    int myid, size;
    char message[100];

    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &myid );
    MPI_Comm_size( MPI_COMM_WORLD, &size );

    if (myid==0) printf("Total size : %5d\n",size);
    //printf("%d\n",myid);
    if ( myid > 3*(size/3) - 1 )
    {
        keyid = MPI_UNDEFINED;
        flag = 0;
    }
    else
    {
        keyid = myid % 3;
        flag = 1;
    }

    MPI_Comm_split( MPI_COMM_WORLD, keyid, myid, &intra_gcomm );
    if ( keyid == 0 )
    {
        MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 1, 1, &inter_lcomm );
        MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 2, 2, &inter_rcomm );
        MPI_Intercomm_merge( inter_lcomm, keyid, &intra_lcomm );
        MPI_Intercomm_merge( inter_rcomm, keyid, &intra_rcomm );
        printf("keyid -%d\n",myid);
    }
    else if ( keyid == 1 )
    {
        MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 0, 1, &inter_rcomm );
        MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 2, 3, &inter_lcomm );
        MPI_Intercomm_merge( inter_rcomm, keyid, &intra_rcomm );
        MPI_Intercomm_merge( inter_lcomm, keyid, &intra_lcomm );
        printf("keyid -%d\n",myid);
    }
    else if (keyid == 2 )
    {
        MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 0, 3, &inter_lcomm );
        MPI_Intercomm_create( intra_gcomm, 0, MPI_COMM_WORLD, 1, 3, &inter_rcomm );
        MPI_Intercomm_merge( inter_lcomm, keyid, &intra_lcomm );
        MPI_Intercomm_merge( inter_rcomm, keyid, &intra_rcomm );
        printf("keyid -%d\n",myid);
    }

    //MPI_Barrier(MPI_COMM_WORLD);
    //printf("barrier-myid %d\n",myid);
    int rrank, lrank, rsize, lsize, gsize, grank;
    if (flag==0)
        printf("process %d is excluded !\n", myid);
    else if (flag==1)
    {
        printf("%d\n",myid);
        //if (myid==0) printf("1\n");
        MPI_Comm_size( intra_lcomm, &lsize );
        MPI_Comm_rank( intra_lcomm, &lrank );
        //if (myid==0) printf("2\n");
```

```
        MPI_Comm_size( intra_rcomm, &rsize );
        MPI_Comm_rank( intra_rcomm, &rrank );
        //if (myid==0) printf("3\n");
        MPI_Comm_size( intra_gcomm, &gsize );
        MPI_Comm_rank( intra_gcomm, &grank );


        if ( myid == 0 )
        {
            printf( "color myid size lrank lsize rrank rsize grank gsize\n" );
            printf( "%5d %4d %4d %5d %5d %5d %5d %5d %5d\n",keyid, myid, size, lrank, lsize, rrank,
            rsize, grank, gsize );
            for ( int i = 1; i < 3*(size/3); i++)
            {
                //mpi_any_source不会堵塞，否则会堵塞
                MPI_Recv( message, 100, MPI_CHAR, MPI_ANY_SOURCE, 10, MPI_COMM_WORLD, &status );
                printf( "%s\n", message );
            }
        }
        else
        {
            printf("%d\n",myid);
            sprintf( message, "%5d %4d %4d %5d %5d %5d %5d %5d %5d",keyid, myid, size, lrank, lsize,
            rrank, rsize, grank, gsize );
            MPI_Send( message, strlen(message)+1, MPI_CHAR, 0, 10, MPI_COMM_WORLD );
        }
    }

    if (myid==0) printf("1\n");
    int ndims=2, cart_rank;
    int dims[2], periods[2], coords[2];
    MPI_Comm comm_cart, comm_new;
    MPI_Comm_split( MPI_COMM_WORLD, flag, myid, &comm_new );
    if (flag==1)
    {
        dims[0] = 3;
        dims[1] = size / 3;
        periods[0] = false;
        periods[1] = false;
        MPI_Cart_create( comm_new, ndims, dims, periods, false, &comm_cart );
        if ( myid == 0 )
        {
            for ( int i = 0; i < 3; i++)
            {
                for(int j = 0; j < size/3; j++)
                {
                    coords[0] = i;
                    coords[1] = j;
                    MPI_Cart_rank( comm_cart, coords, &cart_rank );
                    printf( "{%d, %d} myid=%d\n", coords[0], coords[1], cart_rank );
                }
            }
        }

        MPI_Comm_free( &inter_lcomm );
        MPI_Comm_free( &inter_rcomm );
        MPI_Comm_free( &intra_lcomm );
        MPI_Comm_free( &intra_rcomm );
        MPI_Comm_free( &intra_gcomm );
        MPI_Comm_free( &comm_cart );
        MPI_Comm_free( &comm_new );
    }
    MPI_Finalize();
}
```