Lab 3 Parallel Programming with MPI

Group Communication (1)

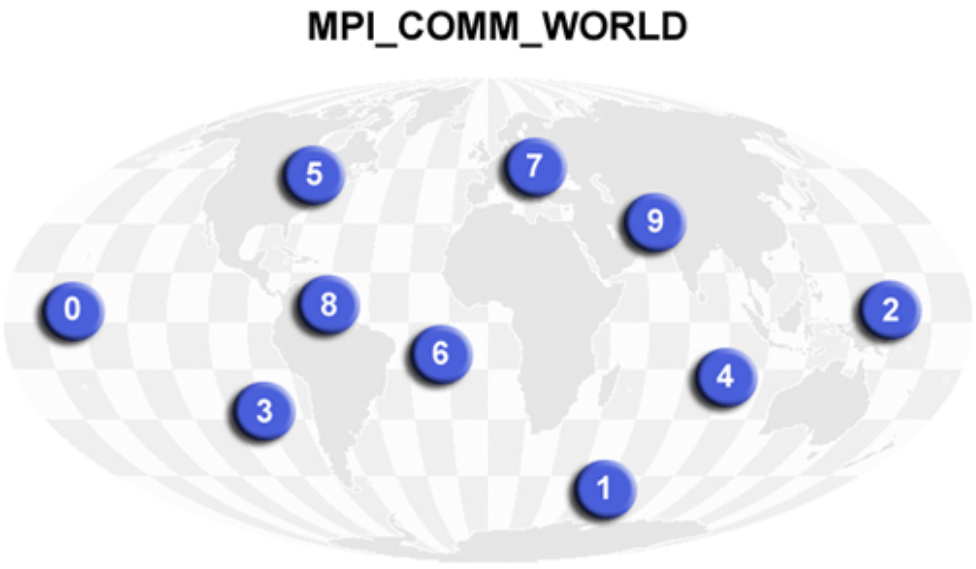
Biên soạn & hướng dẫn: Nguyễn Quang Hùng

1. Mục tiêu

- SV tìm hiểu và sử dụng các hàm collective communication trong thư viện MPI
- Một số hàm giao tiếp nhóm SV cần tìm hiểu :
  - MPI_Bcast(), MPI_Scatter, MPI_Gather(), MPI_Barrier().
  - MPI_Scan(), MPI_Reduce(), MPI_Gatherv(), MPI_Scatterv()…
  - MPI_Reduce_scatter(), MPI_Allreduce …

2. Nội dung

2.1 Giới thiệu



- Sự giao tiếp giữa 1 nhóm process trong cùng communicator
- Mỗi process đều phải gọi hàm giao tiếp nhóm
- SV tìm hiểu xem mỗi hàm giao tiếp nhóm có chức năng gì và thực hiện các chương trình mẫu trong mục 2.2

2.2 Một số chương trình minh họa

2.2.1 Chương trình sử dụng MPI_Barrier():

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

int main(int argc, char **argv){
        int i,rank,size;
        MPI_Init(&argc,&argv);
        MPI_Comm_rank( MPI_COMM_WORLD, &rank);
        MPI_Comm_size( MPI_COMM_WORLD, &size);
        MPI_Barrier( MPI_COMM_WORLD );
        printf("My rank is %d out of %d processes\n",rank,size);
        MPI_Finalize();
        return 0;
}
```

Câu hỏi: Chương trình này có đảm bảo dòng chữ "My rank is …" xuất ra theo đúng thứ tự theo chỉ số rank không? Có cách nào chỉ dùng hàm MPI_Barrier để điều khiển thứ tự xuất của các dòng "My rank is …" đúng thứ tự? Giải thích ngắn gọn.

2.2.2 Chương trình sử dụng MPI_Bcast()

```
#include<mpi.h>

#include <stdio.h>

int main (int argc, char *argv[]) {

        int rank;

        double data;

        MPI_Init(&argc, &argv);

        MPI_Comm_rank(MPI_COMM_WORLD,&rank);

        If (rank==3) data=10.0;

        printf("P:%d before broadcast parameter is %f\n", rank, data);

        MPI_Bcast(&data,1,MPI_DOUBLE, 3, MPI_COMM_WORLD);

        printf("P:%d after broadcast parameter is %f\n", rank, data);

        MPI_Finalize();

        return 0;

}
```

Câu hỏi:

- Chương trình phải chạy với bao nhiêu process thì đúng?
- Kết quả in ra của biến data trên tất cả process bằng bao nhiêu?
- Dùng thêm hàm MPI_Wtime() để đo thời gian thực thi hàm MPI_Bcast() trên các processor. Kết quả có bằng nhau không?

2.2.3 Chương trình sử dụng MPI_Scatter()

```
#include <mpi.h>
#include <stdio.h>
#define N 8
int main( int argc, char* argv[] )
{
        int i;
        int rank, nproc;
        int isend[N], irecv[2];
        MPI_Init( &argc, &argv );
        MPI_Comm_size( MPI_COMM_WORLD, &nproc );
        MPI_Comm_rank( MPI_COMM_WORLD, &rank );
        if(rank == 0) {
            for(i=0; i<N; i++)
               isend[i] = i+1;
        }
        MPI_Scatter( isend, 2, MPI_INT, irecv, 2,MPI_INT, 0, MPI_COMM_WORLD);
        printf("rank = %d, irecv[0] = %d , irecv[1] = %d \n",
              rank, irecv[0], irecv[1]);
        MPI_Finalize();
        return 0;
}
```

Câu hỏi:

- Thực thi chương trình với 4 process và quan sát giá trị xuất ra trên màn hình.
- Sửa lại chương trình này cho phép gửi 5 số nguyên cho 3 process.

2.2.4 Chương trình sử dụng MPI_Gather()

```
#include <mpi.h>

int main( int argc, char* argv[] )
{
int i;
      int rank, nproc;
      int isend, irecv[3];
      MPI_Init( &argc, &argv );
      MPI_Comm_size( MPI_COMM_WORLD, &nproc );
      MPI_Comm_rank( MPI_COMM_WORLD, &rank );
      isend = rank + 1;

      MPI_Gather( &isend, 1, MPI_INT, &irecv, 1,MPI_INT, 0, MPI_COMM_WORLD);
      if(rank == 0) {
      for(i=0; i<3; i++)
      printf("irecv = %d\n", irecv[i]);
      }
      MPI_Finalize();
```

```
}
```

 **THỰC HÀNH**: SV hãy viết các chương trình đơn giản như các ví dụ trên cho các hàm MPI_Reduce, MPI_Scan !

3. Bài tập

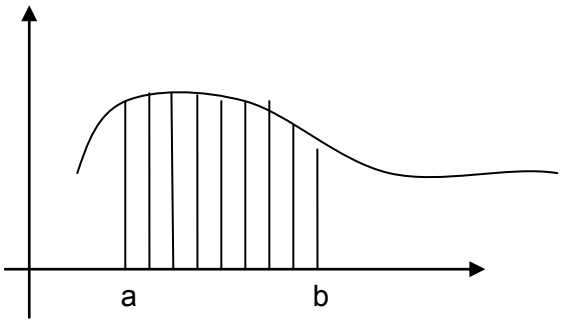
SV hiện thực các bài tập theo hai cách:

a. Sử dụng các hàm truyền thông point to point như MPI_Send, MPI_Recv,…

b. Sử dụng các hàm giao tiếp nhóm.

Bài 1. Viết chương trình nhân hai vector.

Bài 2. Tính tích phân của hàm f(x) > 0 và liên tục trong khoảng [a, b] bằng phương pháp chia miền này thành N=1000000 hình thang nhỏ. Giả sử: hàm:
$f(x) = 4/(1+x^2) > 0$ trong khoảng [0; 1].



LƯU Ý: SV PHẢI NỘP SOURCE CODE CÁC BÀI TẬP LÊN SAKAI ĐÚNG HẠN