Lab 4 Parallel Programming with MPI

Group Communication (2)

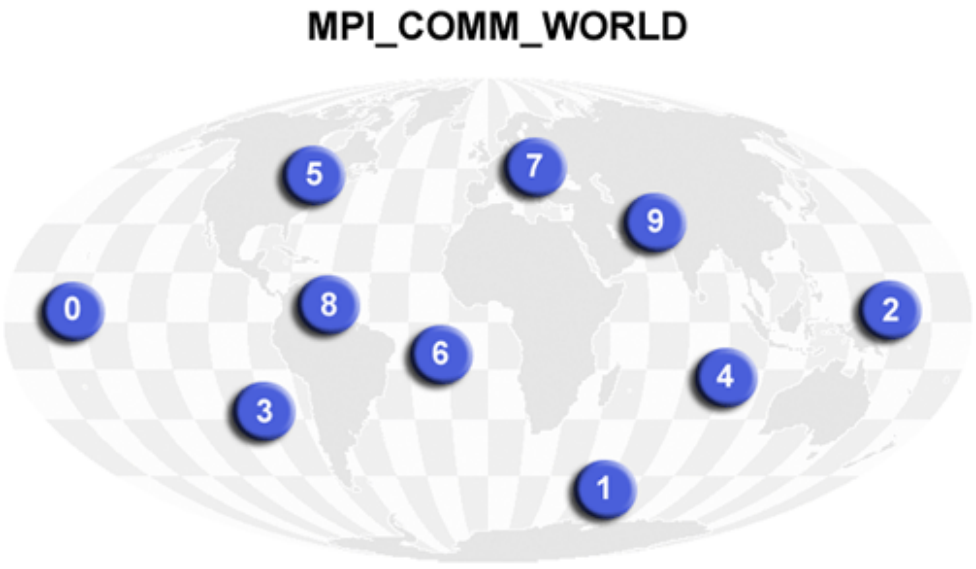
Biên soạn & hướng dẫn: Nguyễn Quang Hùng

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 tính tổng sử dụng MPI_Send() và MPI_Recv():

```
#include <mpi.h>
#include <stdio.h>
#define N 10
float dataArray[N];
float sumLocal(long offset, long chunk);
int main(int argc, char** argv){
    int rank,size;
    long i,offset,chunk;
    float localSum, globalSum;
    MPI_Status status;
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);
    chunk = N/size;
    /* divide data to other processes */
    if(rank == 0) {
        for(i=0; i < N; i++) {
            dataArray[i] = random() % 1000;
            printf("%lf , ", dataArray[i] );
        }
        offset = chunk;
        for(i=1; i < size; i++) {
            MPI_Send(&offset,1,MPI_LONG,i,111,MPI_COMM_WORLD);
            MPI_Send(&dataArray[offset],chunk,MPI_FLOAT,i, 222,MPI_COMM_WORLD);
            offset += chunk;
        }
        /* process 0 execute its work */
        offset = 0;
        localSum = sumLocal(offset,chunk);
        /* process 0 receive resutls from other processes */
        globalSum = localSum;
        for(i=1; i < size; i++){
            MPI_Recv(&localSum, 1, MPI_FLOAT, i, 333, MPI_COMM_WORLD, &status);
            globalSum += localSum;
        }
        fprintf(stdout,"\n The result is : %f \n",globalSum);
    }
    else {
        MPI_Recv(&offset,1, MPI_LONG, 0, 111, MPI_COMM_WORLD, &status);
        MPI_Recv(&dataArray[offset],chunk,MPI_FLOAT,0,222,MPI_COMM_WORLD,&status);
        localSum = sumLocal(offset,chunk);
        MPI_Send(&localSum, 1, MPI_FLOAT, 0, 333, MPI_COMM_WORLD);
    }
    MPI_Finalize();
    return 0;
}

float sumLocal(long offset, long chunk){
    long i;
    float s=0;
```

```
    for(i=offset; i < offset+chunk; i++)
    s += dataArray[i];
    return s;
}
```

Câu hỏi:

1. Chương trình này có giải quyết trường hợp số các giá trị cần tính (*N*) không chia hết cho số (*size*) MPI process cần chạy không? Ví dụ: N=11, có 4 MPI proccess.

2.2.2 SV hãy phát triển chương trình trên dùng hàm giao tiếp nhóm:

```
#include <mpi.h>
#include <stdio.h>
#define N 100000

float dataArray[N];
float sumLocal(float A[], long chunk);
int main(int argc, char** argv){
        int rank,size;
        long i,offset,chunk;
        float localSum, globalSum;
        MPI_Status status;
        MPI_Init(&argc,&argv);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
        MPI_Comm_size(MPI_COMM_WORLD,&size);
        chunk = N/size;
        float bufArray[chunk];
        /* divide data to other processes */
        if(rank == 0)
        {
                for(i=0; i < N; i++)
                        // dataArray[i] = random() % 1000;
                        dataArray[i] = 1;
        }
        /* divide data to every process */
        ...
```

```
        /* compute localSum */
        localSum = sumLocal(bufArray,chunk);
        /* compute globalSum */
        ...
        if(rank == 0) {
                fprintf(stdout,"\n The result is : %f \n",globalSum);
        }
        MPI_Finalize();
        return 0;
}
float sumLocal(float A[], long chunk){
        long i;
        float s=0;
        for(i=0; i < chunk; i++)
                s += A[i];
        return s;
}
```

Câu hỏi:

- Chương trình phải chạy với bao nhiêu process thì đúng?

 **THỰC HÀNH**: SV hãy viết các chương trình đơn giản như các ví dụ trên.

2.3 Hướng dẫn chạy chương trình MPI trên nhiều hosts

Giả sử hệ thống cluster có hai hosts với địa chỉ IP như sau:

gpuserver01: 172.28.10.237

gpuserver02: 172.28.10.236

để chạy chương trình MPI trên hai hosts này, các bạn khai báo sau:

- Trước khi chạy OpenMPI, SV kiểm tra thư viện OpenMPI đã được cài ở hai hosts. Liên hệ GV để biết OpenMPI có thể dùng cho user 'std01' đang dùng không?

Bước 1: tạo file myhosts chứa thông tin tên/địa chỉ IP của các hosts trên cluster cần chạy:

$ vi myhosts

172.28.10.237

172.28.10.236

Bước 2: viết code chương trình MPI, ví dụ: nhanmatranmpi.c

Bước 3: thử biên dịch source code này bằng lệnh:

$ mpicc -o nhanmatranmpi nhanmatranmpi.c -lm

Tham số:

-o: xuất ra tên chương trình thực thi.

-lm: sử dụng thư viện như libm (thư viện gồm các hàm toán học) được cài trong thư mục /usr/lib/

Các SV nên chạy trước chương trình trên máy PC cục bộ trước khi copy source lên server.

Bước 4: copy source 'nhanmatranmpi.c' lên máy server, 172.28.10.237, đặt vào thư mục $HOME

- Trên máy PC, mở terminal mới và gõ lệnh:

$ scp nhanmatranmpi.c std01@172.28.10.237:~/

Bước 5: compile chương trình MPI và copy chương trình thực thi sang các hosts muốn chạy, ví dụ: cả hai hosts liệt kê trong nội dung file 'myhosts' là: 172.28.10.237 và 172.28.10.236

Bước 6: chạy chương trình MPI bằng lệnh sau:

$ mpirun --hostfile myhosts -np 4 /home/std01/testmpi/nhanmatranmpi 1000 1000 1000

Tham số:

--hostfile : sẽ khởi tạo các quá trình MPI trên các hosts được liệt kê trong file 'myhosts'

-np : số process MPI cần tạo ra

3. Bài tập

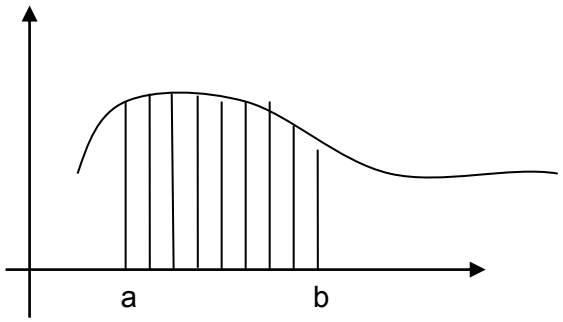
SV hiện thực các bài tập theo hai cách:

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

b. Sử dụng các hàm giao tiếp nhóm.

Bài 3.1. Viết chương trình đếm số lần xuất hiện của số "target" cho trước trong mảng số thực có kích thước là N, với $N > 10^{10}$ và ghi nhận chỉ số các phần tử của mảng có chứa target lên file.

Bài 3.2. 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:
$f(x) = x^2$ trong khoảng [2; 5].
Yêu cầu:
- Viết chương trình tuần tự và nhận xét kết quả.
- Viết chương trình song song hóa bài toán trên (theo 2 cách) .
- Nhận xét về kết quả của chương trình tuần tự và chương trình song song !



Đến bài lab này, sv đã tìm hiểu các hàm cơ bản của MPI và Pthread hãy vận dụng kiến thức đã biết thực hiện bài tập sau:

Bài 3.3. Viết chương trình tính tích hai vector kết hợp cả hai mô hình MPI và multithread.

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