# Lab 4 Parallel Programming with MPI

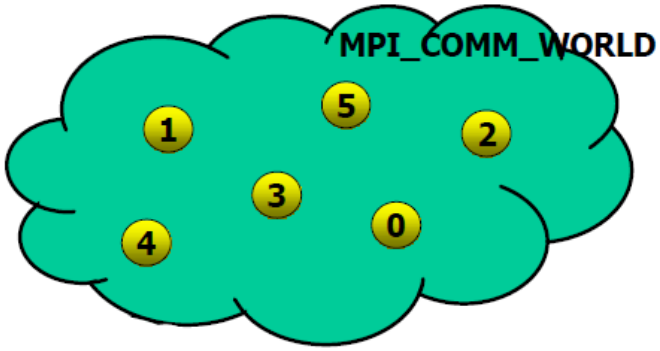
## *Collective communication (2)*

## 1 Mục tiêu

- SV tìm hiểu cách song song hóa chương trình bằng các hàm gửi nhận (send & recv)
- SV  phát triển chương trình đã song song hóa bằng các hàm giao tiếp nhóm.
- Nhận xét về mã nguồn và kết quả của cả 2 chương trình.

## 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

### 2.2 Một số chương trình minh họa

#### 2.2.1 Chương trình tính tổng dùng send và recv:

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

#define N 100000

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;
      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;
}
```

### 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;
}
```

☺ Lưu ý có hai cách để giải bài trên bằng hàm giao tiếp nhóm !

## 3 Bài tập

*SV hiện thực các bài tập theo hai cách:*

*1 Sử dụng các hàm point to point (MPI_Send, MPI_Recv).*

*2 Sử dụng các hàm giao tiếp nhóm.*

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.

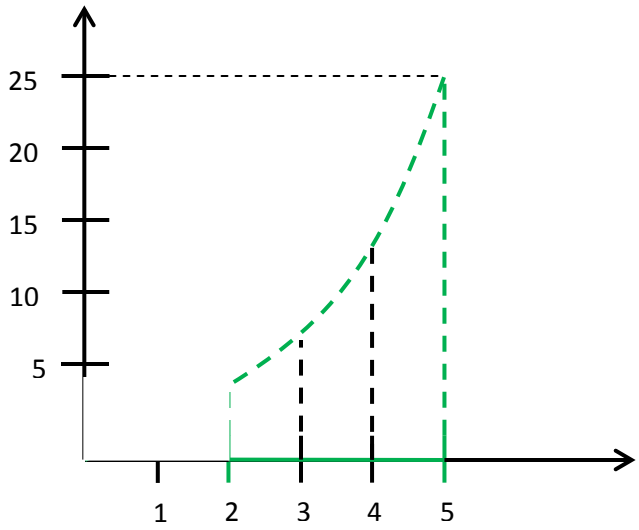
3.2 Viết chương trình tính tích phân sau:

$$\int_2^5 f(x)dx \ , \quad \text{Với } f(x) = x^2$$

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 !

Hướng dẫn:




Tích phân cần tính chính là diện tích hình được giới hạn bởi các đường biên màu xanh lá, SV có thể xấp xỉ diện tích hình này bằng cách tính diện tích các hình chữ nhật nhỏ hơn bên trong. SV thay đổi số lượng hình chữ nhật được chia để đánh giá độ chính xác của giải thuật.

☺ 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:

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