LẬP TRÌNH THƯ VIỆN MPI

MESSAGE PASSING INTERFACE

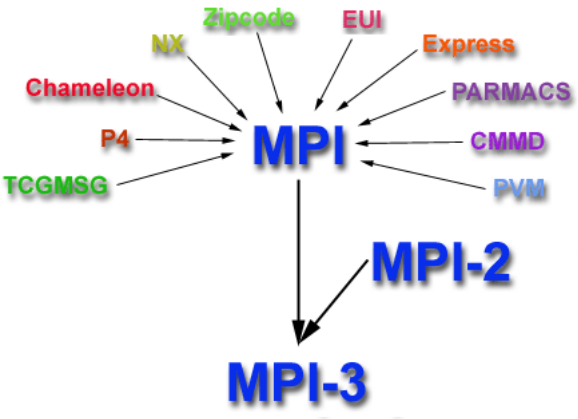
ThS. Phan Trọng Tiến
Bộ môn CNPM – Khoa CNTT
Học Viện Nông nghiệp Việt Nam
Email: phantien84@gmail.com

MPI là gì?

- Là viết tắt của Message Passing Interface, chỉ một dạng giao thức kết nối của máy tính. Nó nằm trong chuẩn "de facto" cho kết nối giữa các nút chạy một chương trình song song trên bộ nhớ phân tán.
- Tập MPI thi hành bao gồm một thư viện các thủ tục sao cho có thể gọi được từ các chương trình Fortran, C, C++ hay Ada.

Mô hình lập trình

- MPI ra đời mục đích dành cho các hệ thống máy tính có bộ nhớ phân tán. Tuy nhiên MPI cũng có thể triển khai được trên hệ thống máy tính có bộ nhớ chia sẻ.



Lập trình MPI

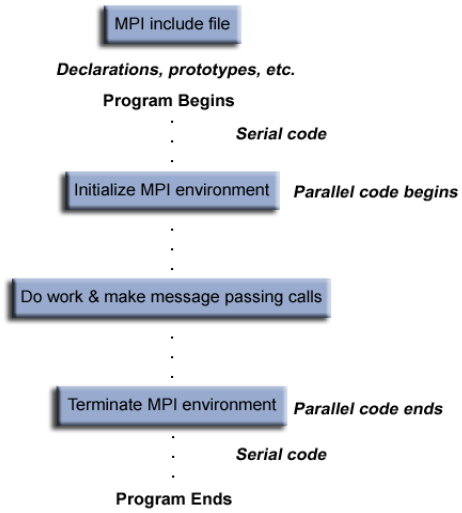
- Header file:
  - Yêu cầu cho mọi chương trình được lập trình bởi MPI
- Định dạng các hàm trong MPI

```
#include “mpi.h”
```

```
rc = MPI_Xxxxx(parameter)
```

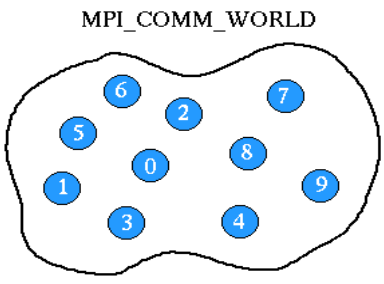
Lập trình MPI

- Cấu trúc của chương trình MPI



Communicators and Groups

- Communicators và Group là tập hợp tất cả các processes có thể giao tiếp được với nhau.
- Phần lớn các hàm trong thư viện MPI yêu cầu tham số Communicator.
- MPI_COMM_WORLD được định nghĩa sẵn.



Rank

- Với mỗi Communicator, mỗi processes có một ID nhất định.
- Rank được bắt đầu từ 0
- Sử dụng rank trong các message để chỉ ra nguồn (source) và đích (destination)

Các hàm quản lý môi trường

- MPI_Init
  - Khởi tạo môi trường thực thi MPI. Hàm này được gọi trong mọi chương trình MPI, được gọi trước các hàm MPI khác, và chỉ được gọi một lần duy nhất.

```
int MPI_Init(int *argc,char ***argv)
```

Các hàm quản lý môi trường

- MPI_Comm_size
  - Xác định số lượng process trong nhóm ứng với một Communicator (thường là MPI_COMM_WORLD)
- MPI_Comm_rank
  - Trả về id của Communicator hiện tại

```
int MPI_Comm_size(MPI_Comm comm,int *size )
int MPI_Comm_rank (comm,&rank)
```


Các hàm quản lý môi trường

- MPI_Abort
  - Hủy bỏ tất cả các MPI processes gắn với một Communicator

```
int MPI_Abort(MPI_Comm comm,int errorcode)
```

Các hàm quản lý môi trường

- MPI_Finalize
  - Kết thúc môi trường thực thi MPI

```
int MPI_Finalize()
```


Ví dụ

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

  int main(int argc, char *argv[])
  {
  int  numtasks, rank, rc;

  rc = MPI_Init(&argc,&argv);
  if (rc != MPI_SUCCESS) {
    printf ("Error starting MPI program. Terminating.\n");
    MPI_Abort(MPI_COMM_WORLD, rc);
    }

  MPI_Comm_size(MPI_COMM_WORLD,&numtasks);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  printf ("Number of tasks= %d My rank= %d\n", numtasks,rank);

  /*******  do some work *******/

  MPI_Finalize();
  }
```

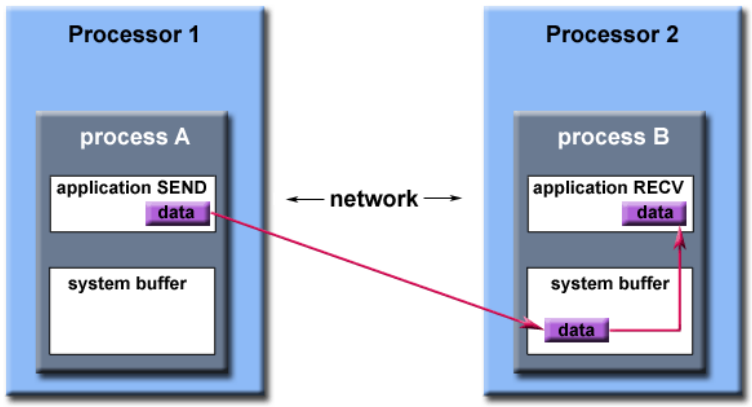
Giao tiếp Point to Point

- MPI Point to Point là giao tiếp giữa hai và chỉ hai processes với nhau.
- Khi một Process thực hiện giao thức truyền tin (send) thì nhiệm vụ khác phải có giao thức nhận tin (receive) tương ứng.

Giao tiếp Point to Point

- Các kiểu truyền và nhận trong MPI
  - Synchronous send
  - Blocking send/blocking receive
  - Non-Blocking send/ non-blocking receive
  - Buffered send
  - Combined send/receive
  - Ready send

Bộ nhớ đệm (Buffer)



Path of a message buffered at the receiving process

Blocking

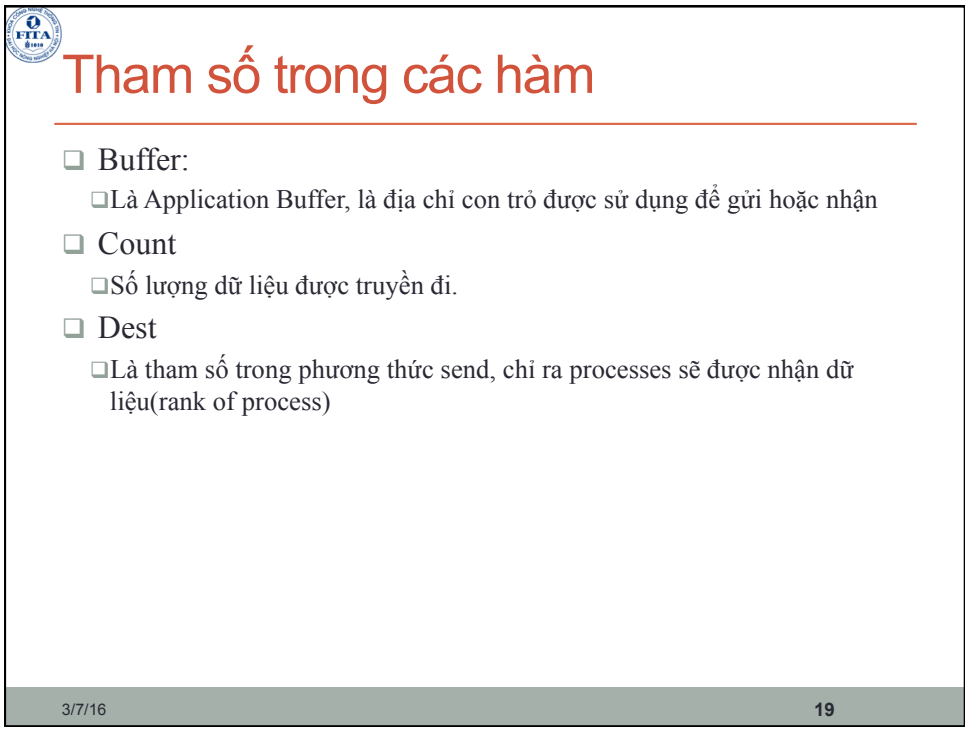
- Phương thức send sẽ kết thúc khi application buffer được truyền đi.
- Phương thức send đồng bộ (synchronous) khi bên nhận (receive) xác nhận thành công.
- Phương thức send không đồng bộ nếu system buffer được dùng để lưu dữ liệu.
- Phương thức receive sẽ kết thúc sau khi nhận được dữ liệu.

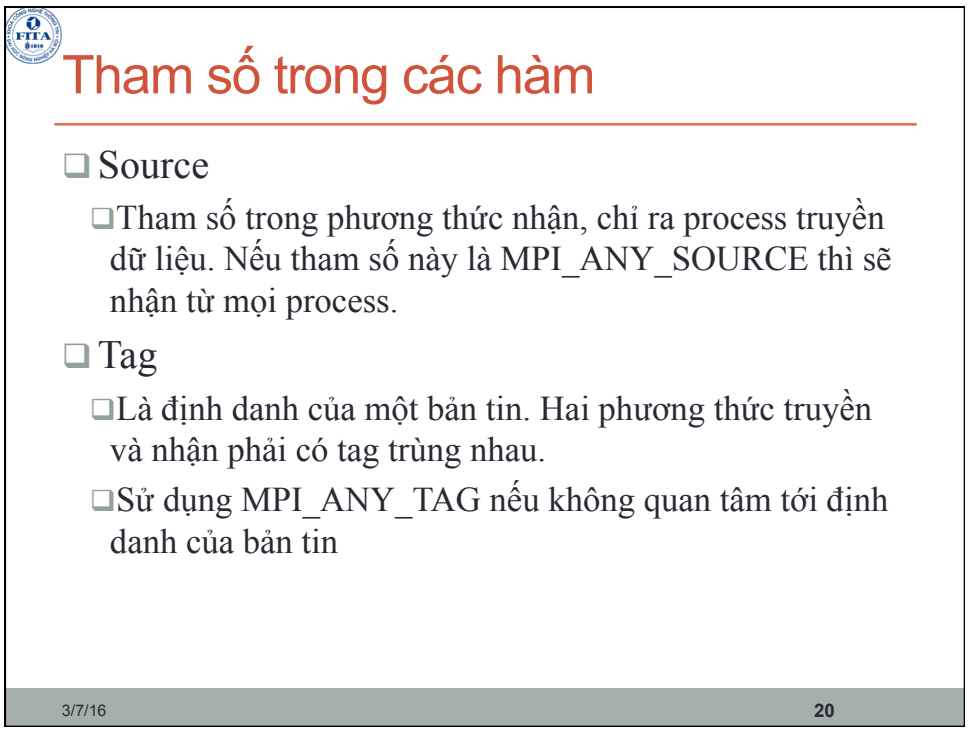
Non-blocking

- Hai phương thức send và receive có hoạt động giống nhau, chúng sẽ kết thúc ngay sau lời gọi mà không quan tâm dữ liệu đã gửi hoặc nhận xong hay chưa.
- Cần sử dụng phương thức "wait"

Các hàm trong Point-to-Point

Blocking sends	MPI_Send(buffer, count, type, dest, tag, comm)
Non-Blocking sends	MPI_Isend(buffer, count, type, dest, tag, comm, request)
Blocking receive	MPI_Recv(buffer, count, type, source, tag, comm, status)
Non-blocking receive	MPI_Irecv(buffer, count, type, source, tag, comm, request)


Tham số trong các hàm
Buffer:
Là Application Buffer, là địa chỉ con trỏ được sử dụng để gửi hoặc nhận
Count
Số lượng dữ liệu được truyền đi.
Dest
Là tham số trong phương thức send, chỉ ra processes sẽ được nhận dữ liệu(rank of process)
3/7/16
19


Tham số trong các hàm
Source
Tham số trong phương thức nhận, chỉ ra process truyền dữ liệu. Nếu tham số này là MPI_ANY_SOURCE thì sẽ nhận từ mọi process.
Tag
Là định danh của một bản tin. Hai phương thức truyền và nhận phải có tag trùng nhau.
Sử dụng MPI_ANY_TAG nếu không quan tâm tới định danh của bản tin
3/7/16
20

Tham số trong các hàm

- Communicator
  - Chỉ ra communicator mà các processes sẽ giao tiếp với nhau.
- Status
  - Sử dụng trong phương thức nhận, chỉ ra source và tag của bản tin.
  - Được định nghĩa trong cấu trúc MPI_Status (stat.MPI_SOURCE, stat.MPI_TAG)

Tham số trong các hàm

- Request:
  - Được sử dụng trong phương thức send và receive non-blocking.
  - Hệ thống sẽ gán cho mỗi phương thức một "số request" duy nhất để người lập trình sử dụng.

MPI Data type	C Datatype
MPI_CHAR	signed char
MPI_SHORT	signed short in
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_BYTE	
MPI_PACKED	

MPI_Send

- Là phương thức blocking send. Hàm này sẽ kết thúc khi application buffer gửi hết dữ liệu.

```
MPI_Send(&buf,count,datatype,dest,tag,comm)
```

MPI_Recv

- Là phương thức blocking Receive, kết thúc khi dữ liệu trong application buffer được nhận.

```
MPI_Recv(&buf,count,datatype,source,tag,comm,&status)
```


MPI_Ssend

- Là phương thức blocking send đồng bộ. Nó sẽ kết thúc khi application buffer được truyền đi hết và bên nhận bắt đầu nhận dữ liệu.

```
MPI_Ssend(&buf,count,datatype,dest,tag,comm)
```

MPI_Isend

- Hàm sẽ kết thúc mà không quan tâm tới application buffer đã được gửi hết hay chưa. Do đó, chương trình không nên thay đổi giá trị của application buffer.
- Sử dụng hàm MPI_Wait hoặc MPI_Test để đảm bảo việc truyền dữ liệu thành công

```
MPI_Isend(&buf,count,datatype,dest,tag,comm,&request)
```


MPI_Irecv

- Hàm sẽ kết thúc mà không quan tâm tới việc applicaton buffer đã nhận hết hay chưa.
- Sử dụng MPI_Wait hoặc MPI_Test để kiểm tra việc nhận dữ liệu thành công.

```
MPI_Irecv(&buf,count,datatype,source,tag,comm,&request)
```

MPI_Wait

- Được sử dụng để cho non-blocking send và receive, nó sẽ dừng lại cho đến khi việc nhận và gửi thành công.

```
MPI_Wait (&request,&status)
```

```
MPI_Waitall(count,&array_of_requests,&array_of_statuses)
```

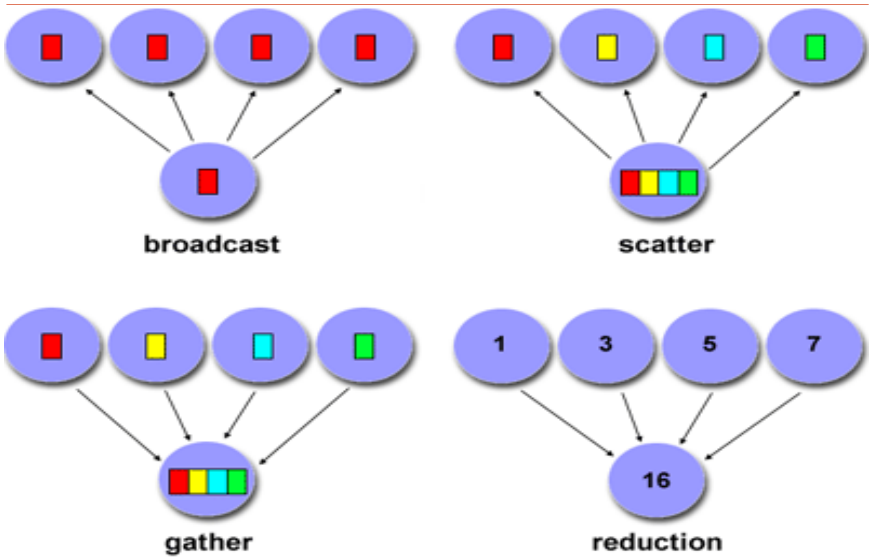

Truyền thông tập hợp

- Các processes có cùng một communicator thì mới giao tiếp được với nhau
- Mặc định, các processes đều có communicator là MPI_COMM_WORLD
- Tùy vào mục đích lập trình để chia các processes theo từng nhóm riêng biệt.

Các kiểu giao tiếp tập hợp

- Đồng bộ:
  - Các processes trong nhóm sẽ chờ nhau.
- Di chuyển dữ liệu
  - Broadcast, scatter/gather, all to all
- Tính toán tập hợp (reduction)
  - Một process trong nhóm sẽ tập hợp dữ liệu và tính toán trên dữ liệu đó

Di chuyển dữ liệu



MPI_Barrier

- Để đồng bộ tất các các process trong nhóm
- Khi mỗi process tới lời gọi MPI_Barrier sẽ dừng lại khi tất cả các process khác đều tới lời gọi MPI_Barrier.

```
MPI_Barrier (comm)
```


MPI_Bcast

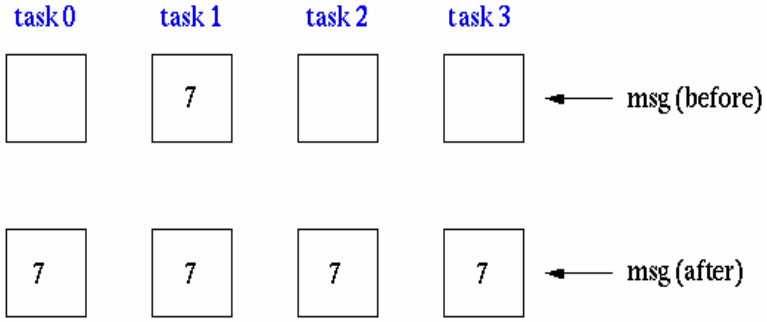
- Gửi một bản tin từ một processes có rank là "root" tới tất cả các process khác

```
MPI_Bcast(&buffer,count,datatype,root,comm)
```

MPI_Bcast

Broadcasts a message to all other processes of that group

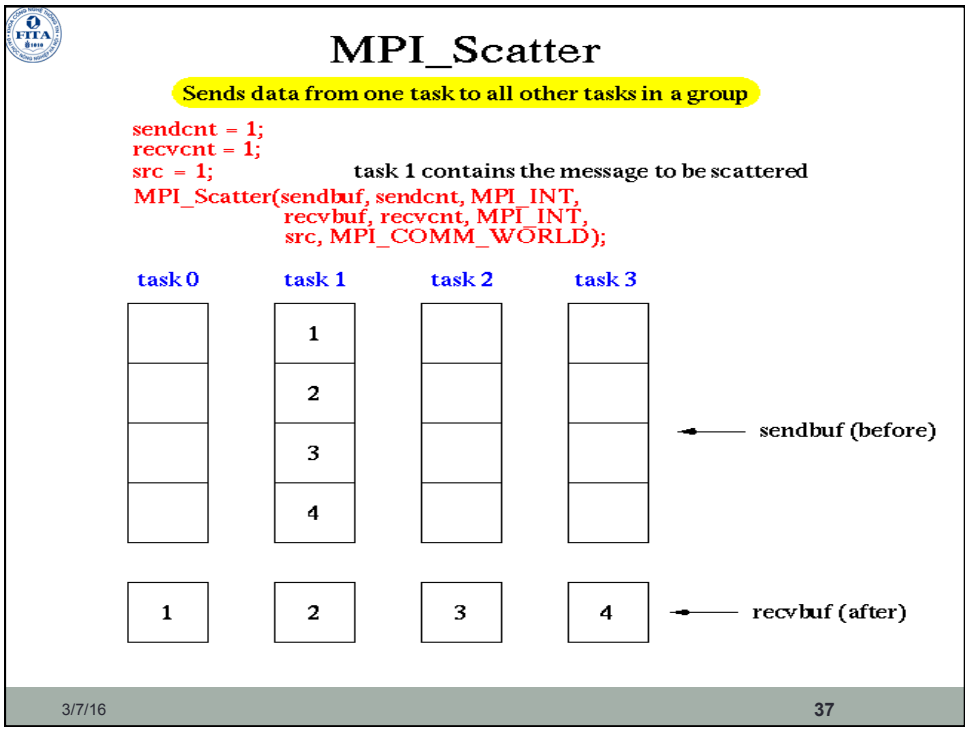
```
count = 1;
source = 1;                    broadcast originates in task 1
MPI_Bcast(&msg, count, MPI_INT, source, MPI_COMM_WORLD);
```

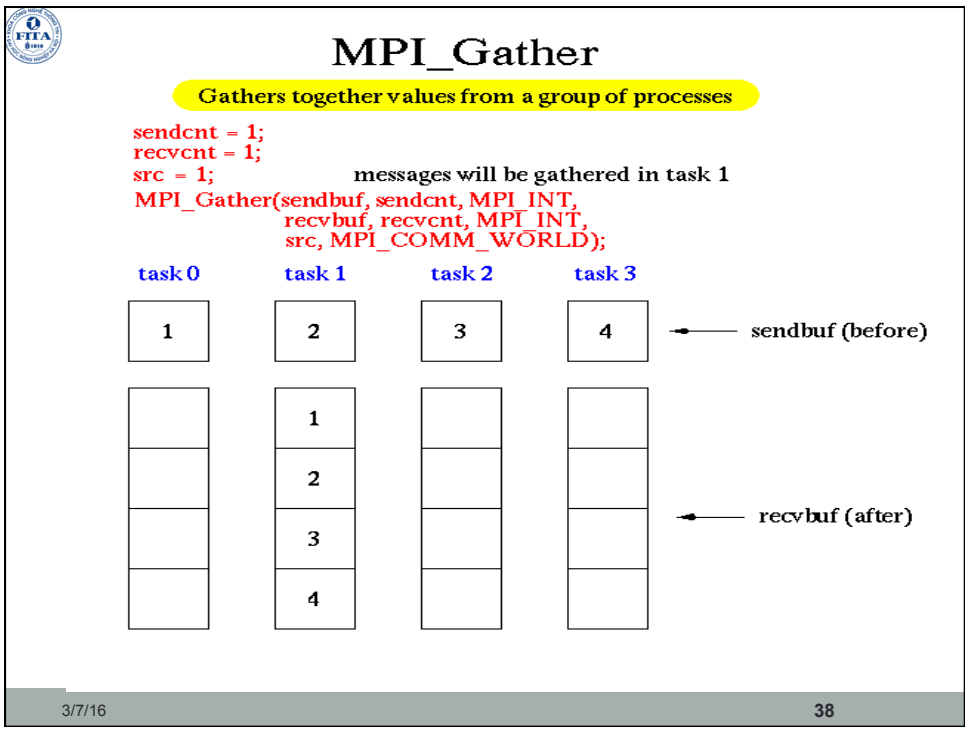


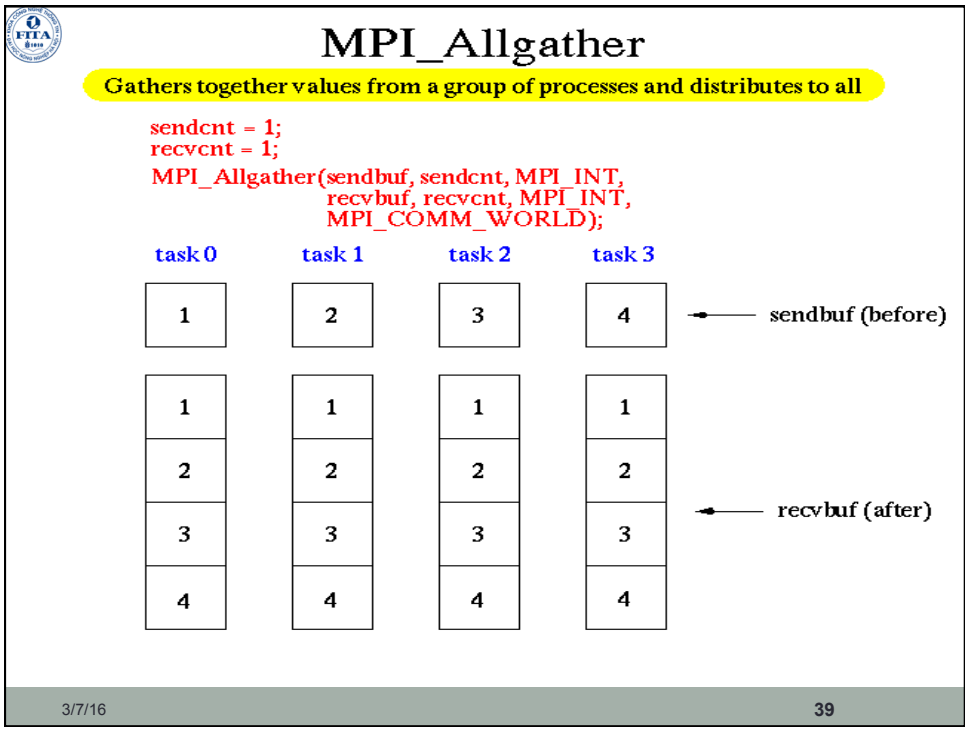
MPI_Scatter

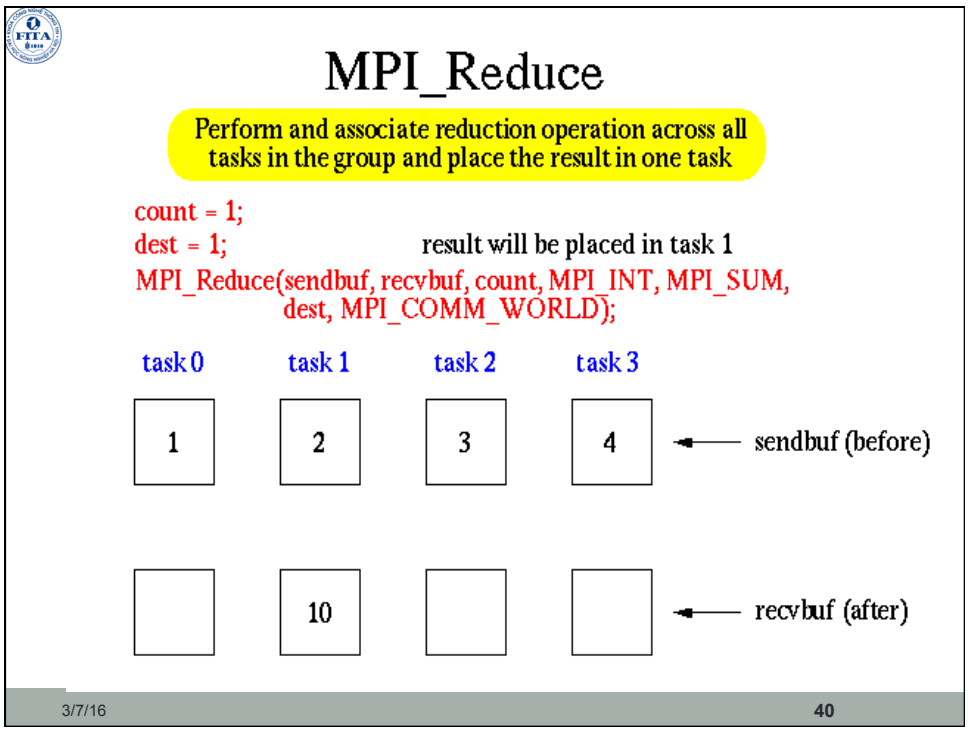
- Gửi dữ liệu từ một process tới các process khác trong nhóm.

```
MPI_Scatter(&sendbuf,sendcnt,sendtype,&recvbuf,recvcnt,recvtype,root,comm)
```


MPI_Scatter
Sends data from one task to all other tasks in a group
sendcnt = 1;
recvcnt = 1;
src = 1;
task 1 contains the message to be scattered
MPI_Scatter(sendbuf, sendcnt, MPI_INT,
recvbuf, recvcnt, MPI_INT,
src, MPI_COMM_WORLD);
task 0
task 1
task 2
task 3
1
2
3
4
sendbuf (before)
1
2
3
4
recvbuf (after)
3/7/16
37


MPI_Gather
Gathers together values from a group of processes
sendcnt = 1;
recvcnt = 1;
src = 1;
messages will be gathered in task 1
MPI_Gather(sendbuf, sendcnt, MPI_INT,
recvbuf, recvcnt, MPI_INT,
src, MPI_COMM_WORLD);
task 0
task 1
task 2
task 3
1
2
3
4
sendbuf (before)
1
2
3
4
recvbuf (after)
3/7/16
38


MPI_Allgather
Gathers together values from a group of processes and distributes to all
sendcnt = 1;
recvcnt = 1;
MPI_Allgather(sendbuf, sendcnt, MPI_INT,
recvbuf, recvcnt, MPI_INT,
MPI_COMM_WORLD);
task 0
task 1
task 2
task 3
1
2
3
4
sendbuf (before)
recvbuf (after)
3/7/16
39

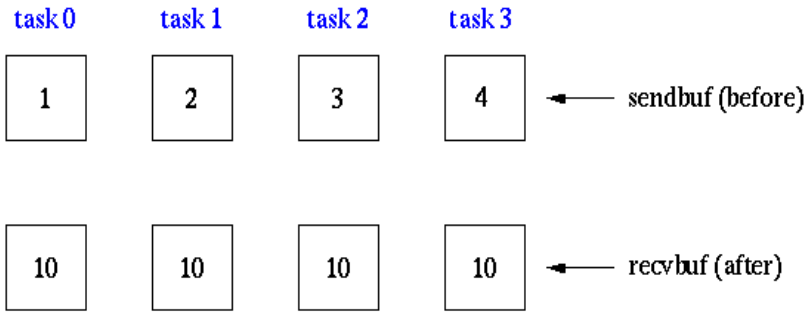

MPI_Reduce
Perform and associate reduction operation across all tasks in the group and place the result in one task
count = 1;
dest = 1;
result will be placed in task 1
MPI_Reduce(sendbuf, recvbuf, count, MPI_INT, MPI_SUM,
dest, MPI_COMM_WORLD);
task 0
task 1
task 2
task 3
1
2
3
4
sendbuf (before)
10
recvbuf (after)
3/7/16
40

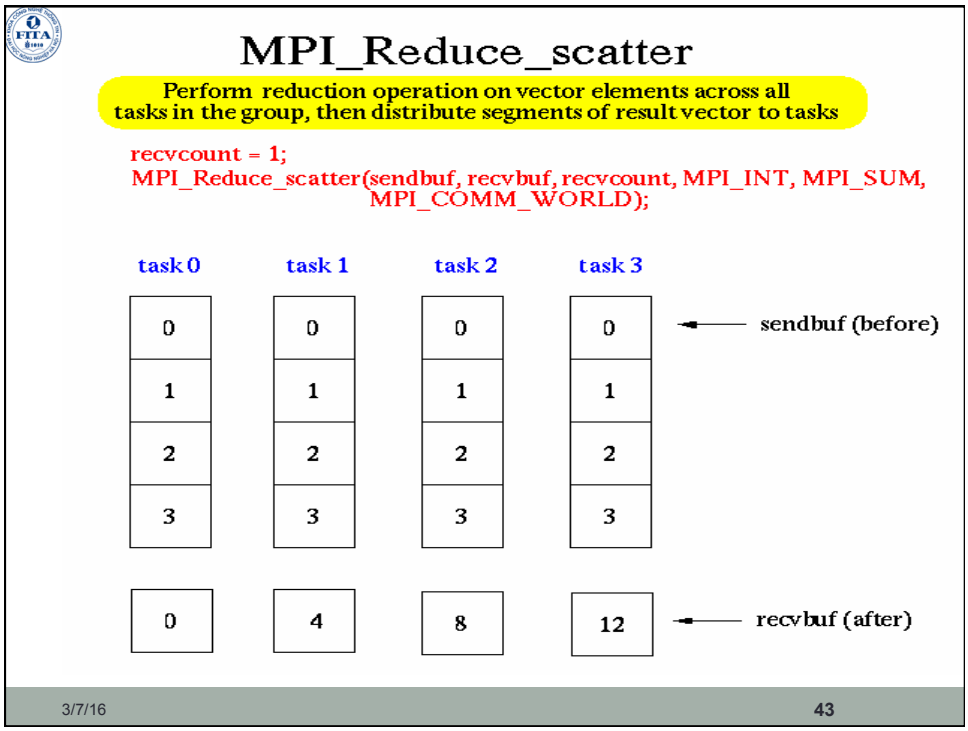
MPI Reduction Operation		C Data types
MPI_MAX	Maximum	Integer,float
MPI_MIN	Minimum	Integer,float
MPI_SUM	Sum	Integer,float
MPI_PROD	Product	Integer,float
MPI_LAND	Logical AND	Integer
MPI_BAND	Bit-wise AND	Integer,MPI_BYTE
MPI_LOR	Logical OR	Integer
MPI_BOR	Bit-wise OR	Integer,MPI_BYTE
MPI_LXOR	Logical XOR	Integer
MPI_BXOR	Bit-wise XOR	Integer,MPI_BYTE
MPI_MAXLOC	Max value and location	Float, double and long double
MPI_MINLOC	Min value and location	Float, double and long double

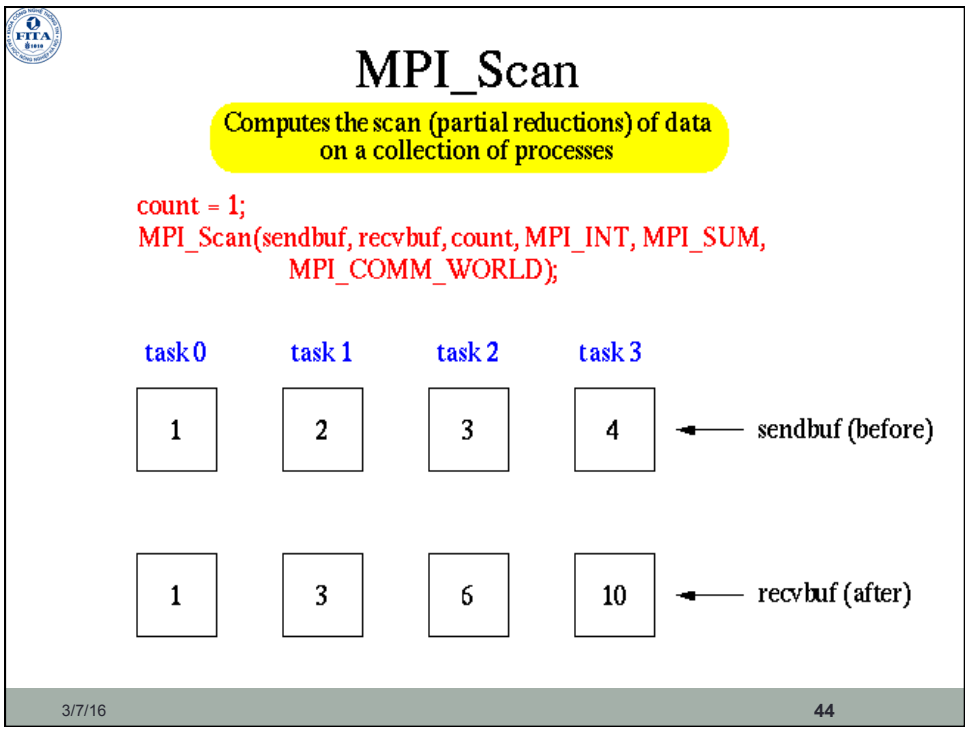
MPI_Allreduce

Perform and associate reduction operation across all tasks in the group and place the result in all tasks

```
count = 1;
MPI_Allreduce(sendbuf, recvbuf, count, MPI_INT, MPI_SUM,
              MPI_COMM_WORLD);
```




MPI_Reduce_scatter
Perform reduction operation on vector elements across all tasks in the group, then distribute segments of result vector to tasks
recvcount = 1;
MPI_Reduce_scatter(sendbuf, recvbuf, recvcount, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
task 0
task 1
task 2
task 3
0 1 2 3
0 1 2 3
0 1 2 3
0 1 2 3
sendbuf (before)
0
4
8
12
recvbuf (after)
3/7/16
43


MPI_Scan
Computes the scan (partial reductions) of data on a collection of processes
count = 1;
MPI_Scan(sendbuf, recvbuf, count, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
task 0
task 1
task 2
task 3
1
2
3
4
sendbuf (before)
1
3
6
10
recvbuf (after)
3/7/16
44