Message Passing Interface. (MPI)

Parallel —

Distributed —

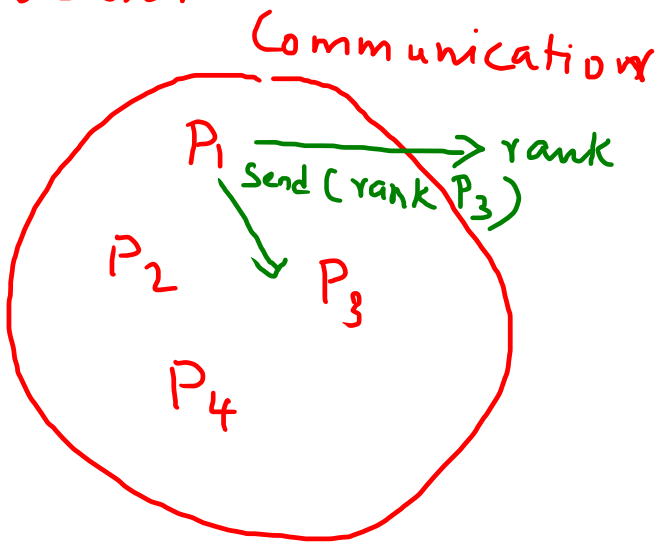
Interconnection Networks —

Message Passing

↳ Point-to-point Comm^n ($P_1 \rightleftarrows P_2$)

↳ Collective Comm^n ($P_1 \rightrightarrows$)

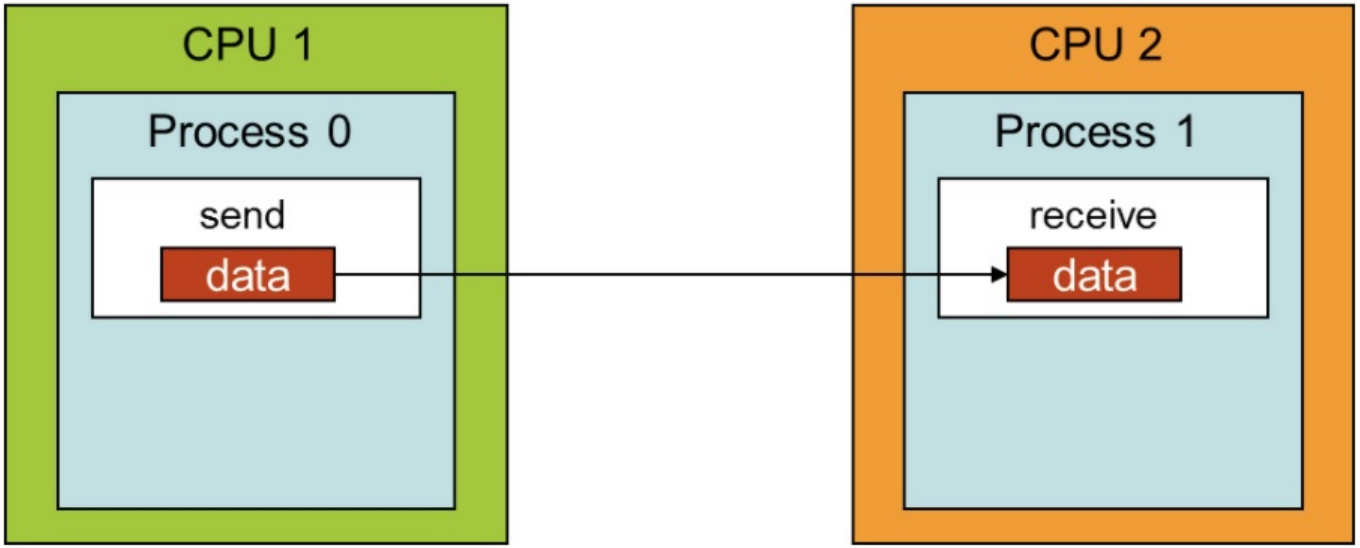
Communicator



0 to n-1

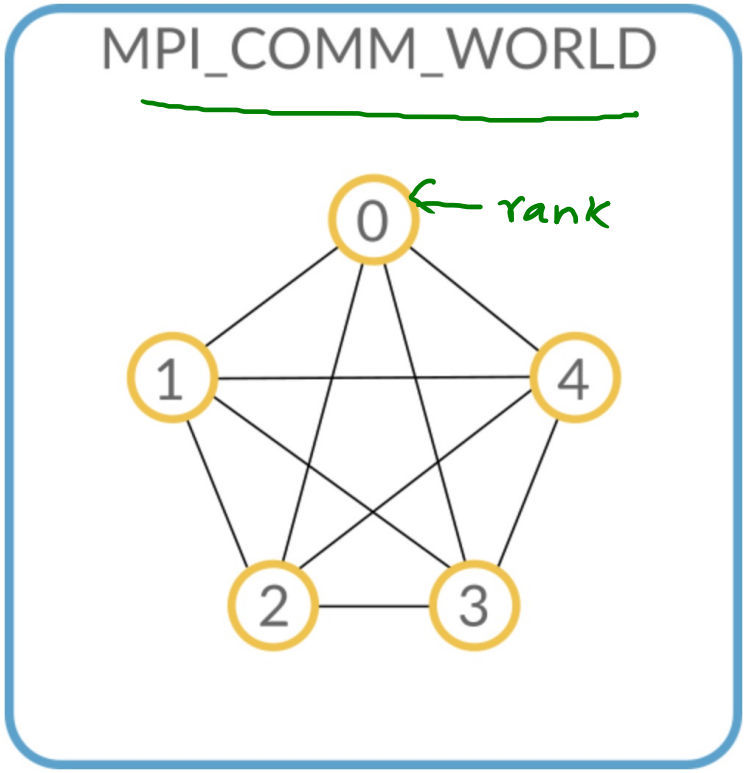
✓ MPI_COMM_WORLD

Point - to - Point Communication



Communicator

Communicator



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

int main(int argc, char **argv)
{
    // Initialize the MPI environment
    MPI_Init(&argc, &argv);

    // Get the number of processes ssociated with the communicator
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of the calling process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    printf("Hello world from process %s with rank %d out of %d processors\n", processor_name, world_rank, world_size);

    // Finalize: Any resources allocated for MPI can be freed
    MPI_Finalize();
}
```

mpirun -n 5 a.out

```
#include <stdio.h>
#include <string.h>
#include <mpi.h>
int main(int argc, char* argv[]){
        /*No MPI call before this*/
        MPI_Init(&argc,&argv);
        ...
        MPI_Finalize();
        /*No MPI call after this*/
        ...
        return 0;
}
```

sequential

NULL NULL

sequential

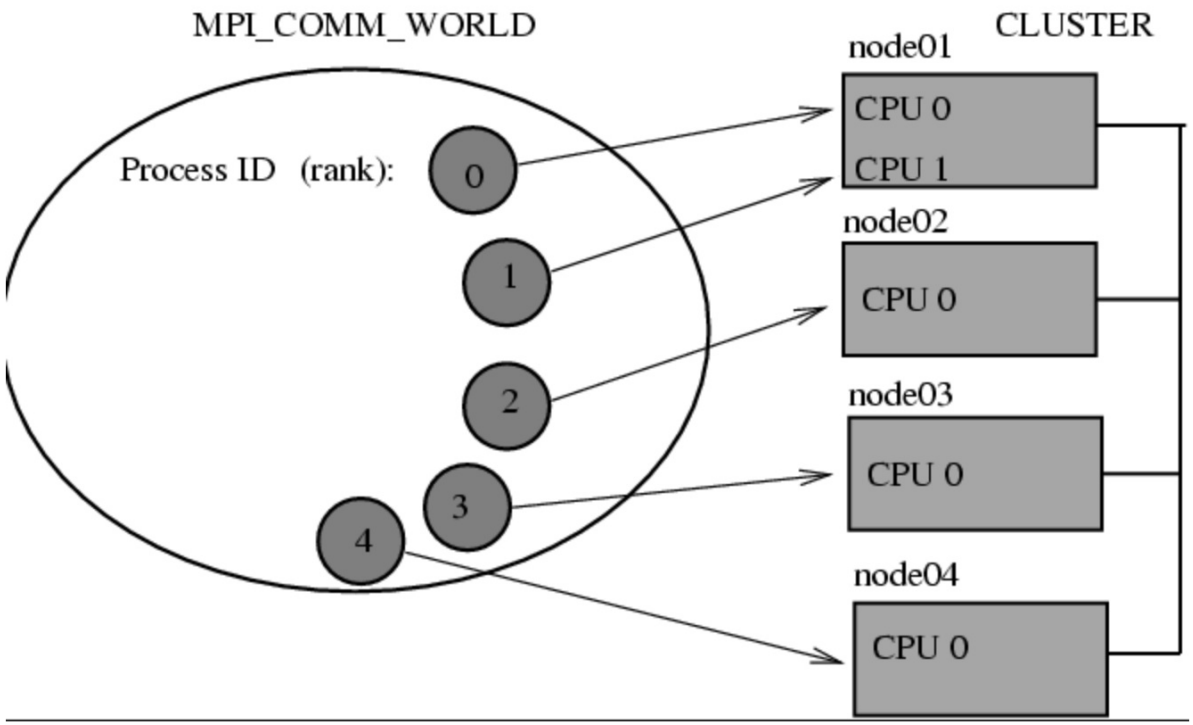
```
int MPI_Comm_size(
        MPI_Comm comm /*in*/,
        int* comm_sz_p /*out*/);

int MPI_Comm_rank(
        MPI_Comm comm /*in*/,
        int* my_rank /*out*/);
```

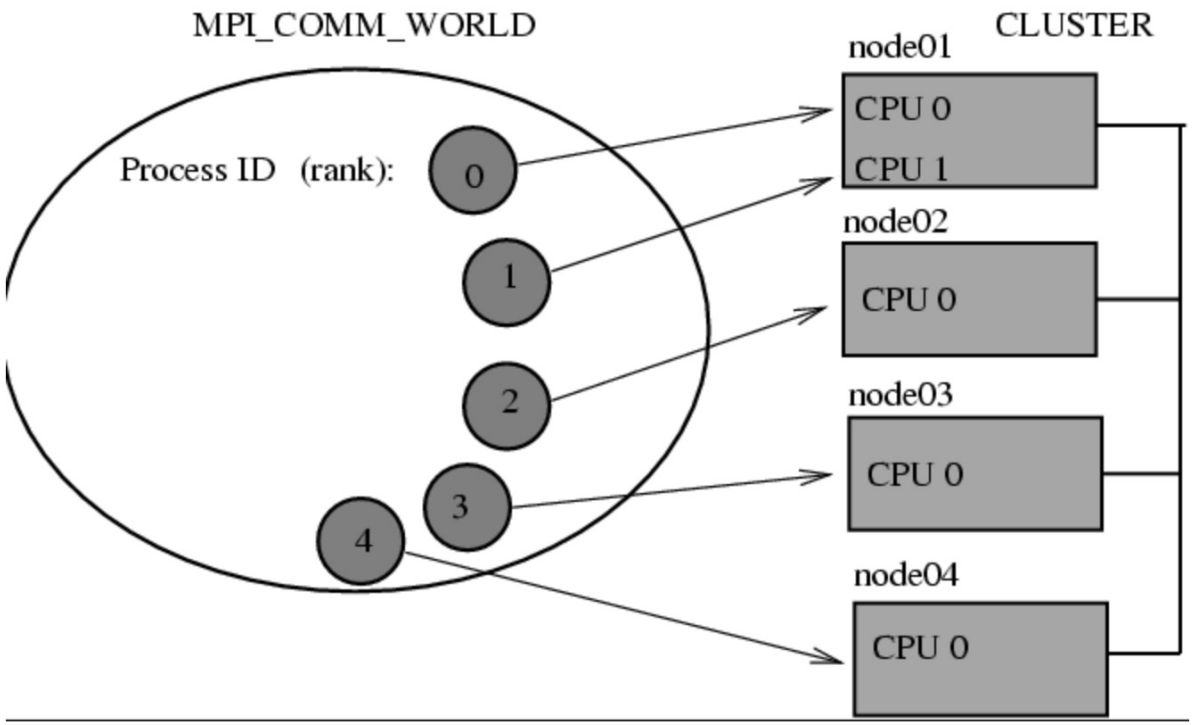
5

OpenMPI

The MPI_COMM_WORLD



The MPI_COMM_WORLD



Point-to Point

Blocking

Send :-

MPI_Send ( )

→ message received by the destination

receive :-

MPI_Recv ( )

Non-blocking

✓ When the msg is send by the sender

"MPI_ANY_TAG"

MPI_Send(buff,
{size,}
datatype
destination
int {tag} 1111
comm)

MPI_Probe() function

find the size of the msg

(source, → MPI_ANY_SOURCE
msg tag – MPI_ANY_TAG
world
&status)

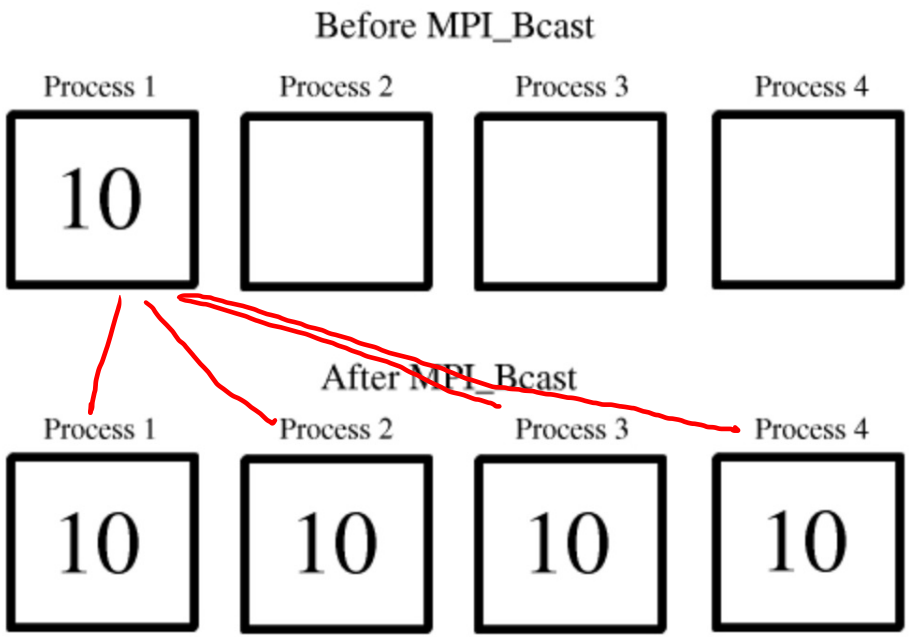
MPI_Recv(.... , 1111, ..)

Collective Communication / Group Communication

MPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm);

Parameter	Meaning of Parameter
buffer	starting address of buffer (choice)
count	number of entries in buffer (integer)
datatype	datatype of buffer (handle)
root	rank of broadcast root (integer)
comm	communicator (handle)

MPI_Bcast broadcasts a message from the process with rank "root" to all other processes of the group.

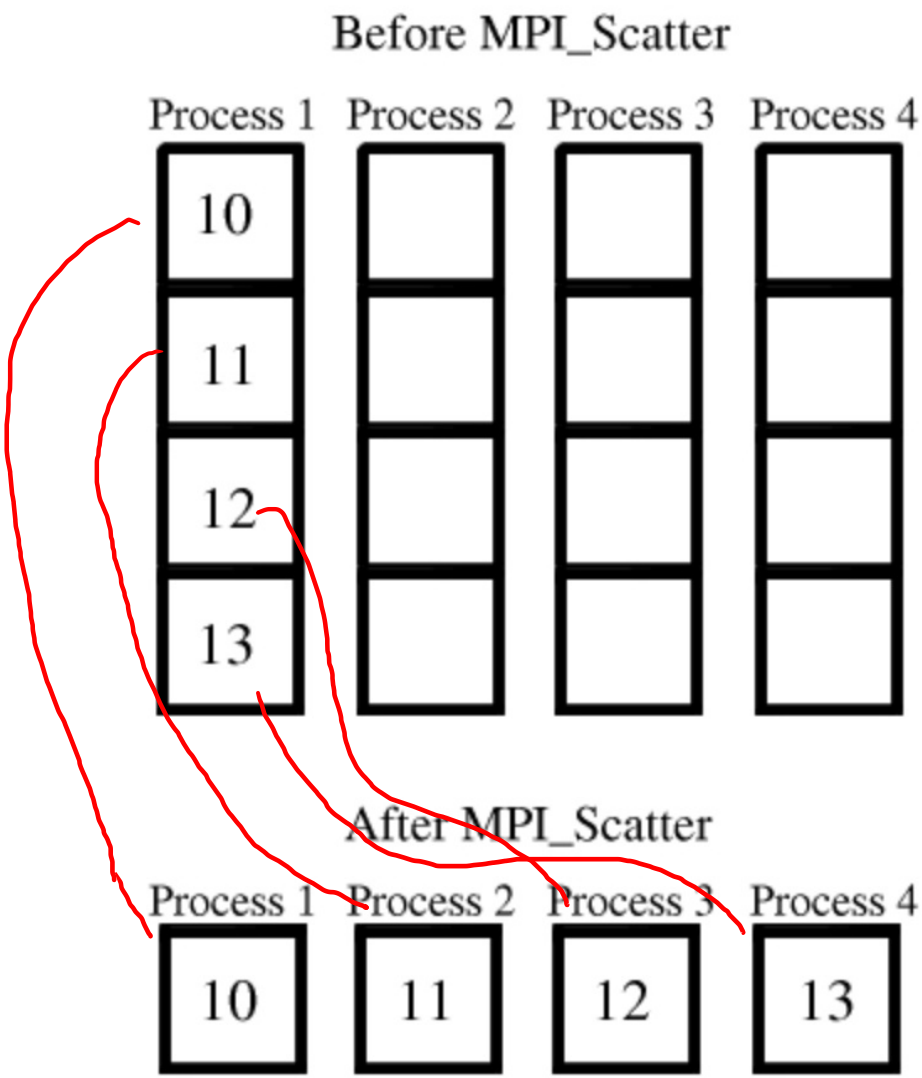

Before MPI_Bcast
Process 1
Process 2
Process 3
Process 4
10
After MPI_Bcast
Process 1
Process 2
Process 3
Process 4
10
10
10
10

MPI_Scatter(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf, int recvcnt, MPI_Datatype recvtype, int root, MPI_Comm comm);

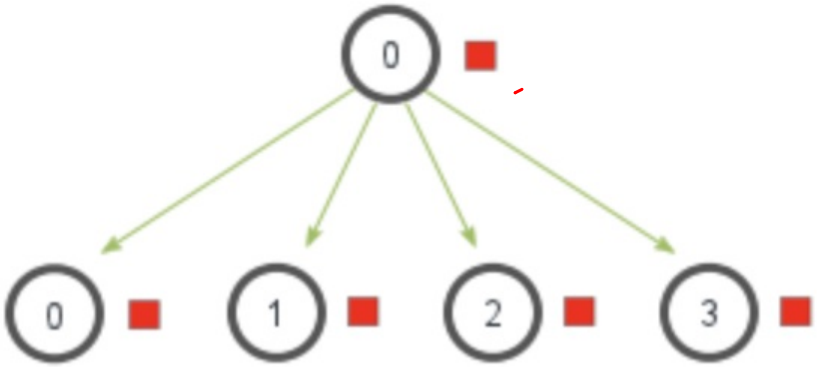
Parameter	Meaning of Parameter
sendbuf	address of send buffer (choice, significant only at root)
sendcnt	number of elements sent to each process (integer, significant only at root)
sendtype	data type of send buffer elements (significant only at root) (handle)
recvbuf	address of receive buffer (choice)
recvcnt	number of elements in receive buffer (integer)
recvtype	data type of receive buffer elements (handle)
root	rank of sending process (integer)
comm	communicator (handle)

MPI_Scatter sends data from one task to all other tasks in a group.

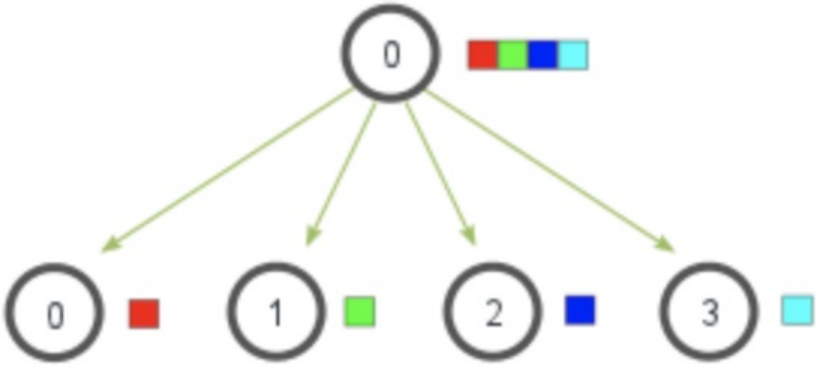
Broad cast — Bcast


Before MPI_Scatter
Process 1
Process 2
Process 3
Process 4
10
11
12
13
After MPI_Scatter
Process 1
Process 2
Process 3
Process 4
10
11
12
13

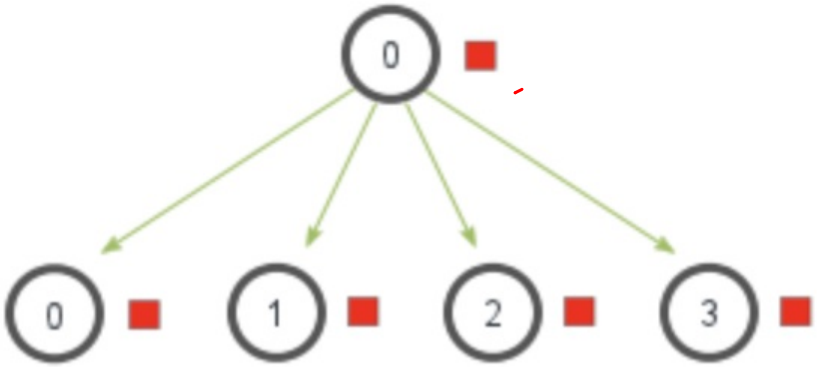
MPI_Bcast



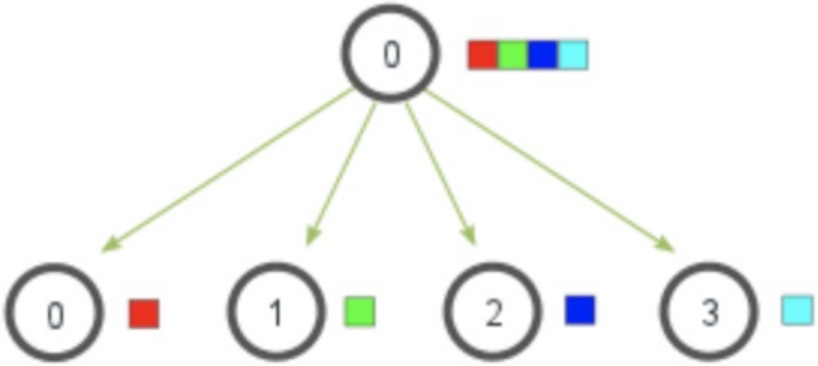
MPI_Scatter



MPI_Bcast



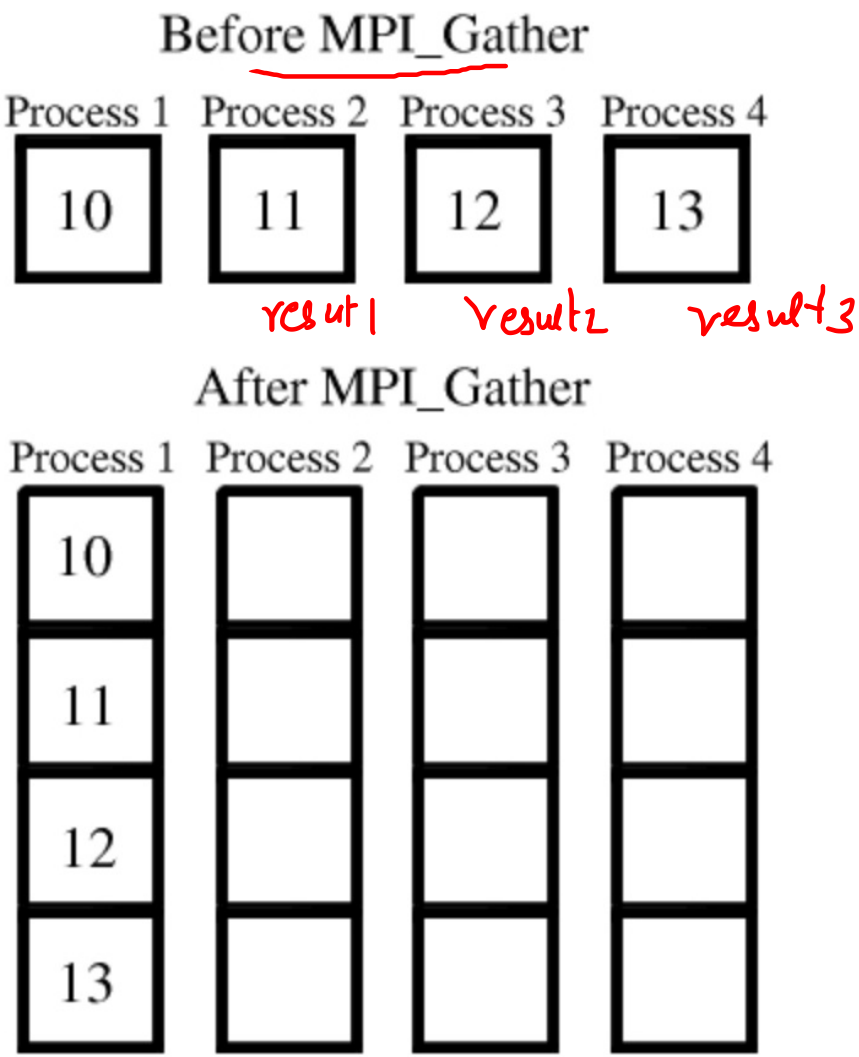
MPI_Scatter



MPI_Gather(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm);

Parameter	Meaning of Parameter
sendbuf	starting address of send buffer (choice)
sendcount	number of elements in send buffer (integer)
sendtype	data type of send buffer elements (handle)
recvbuf	address of receive buffer (choice, significant only at root)
recvcount	number of elements for any single receive (integer, significant only at root)
recvtype	data type of receive buffer elements (significant only at root) (handle)
root	rank of receiving process (integer)
comm	communicator (handle)

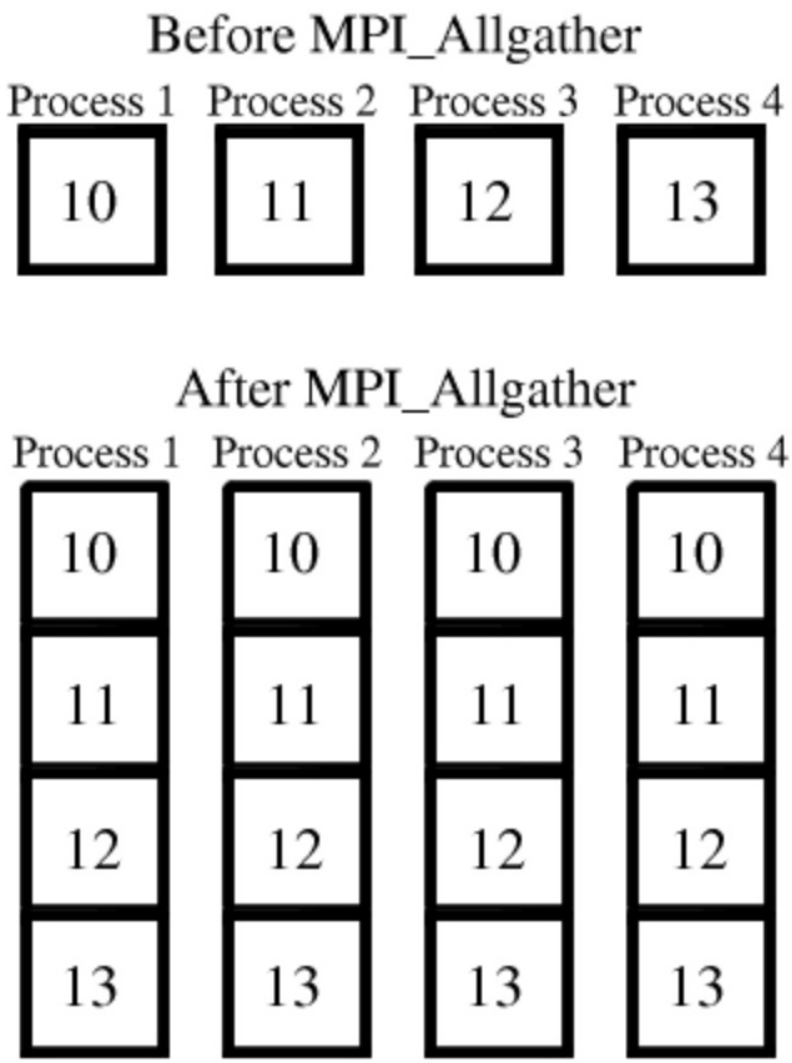
MPI_Gather gathers together values from a group of processes.


Before MPI_Gather
Process 1
Process 2
Process 3
Process 4
10
11
12
13
result1
result2
result3
After MPI_Gather
Process 1
Process 2
Process 3
Process 4
10
11
12
13

MPI_Allgather(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm);

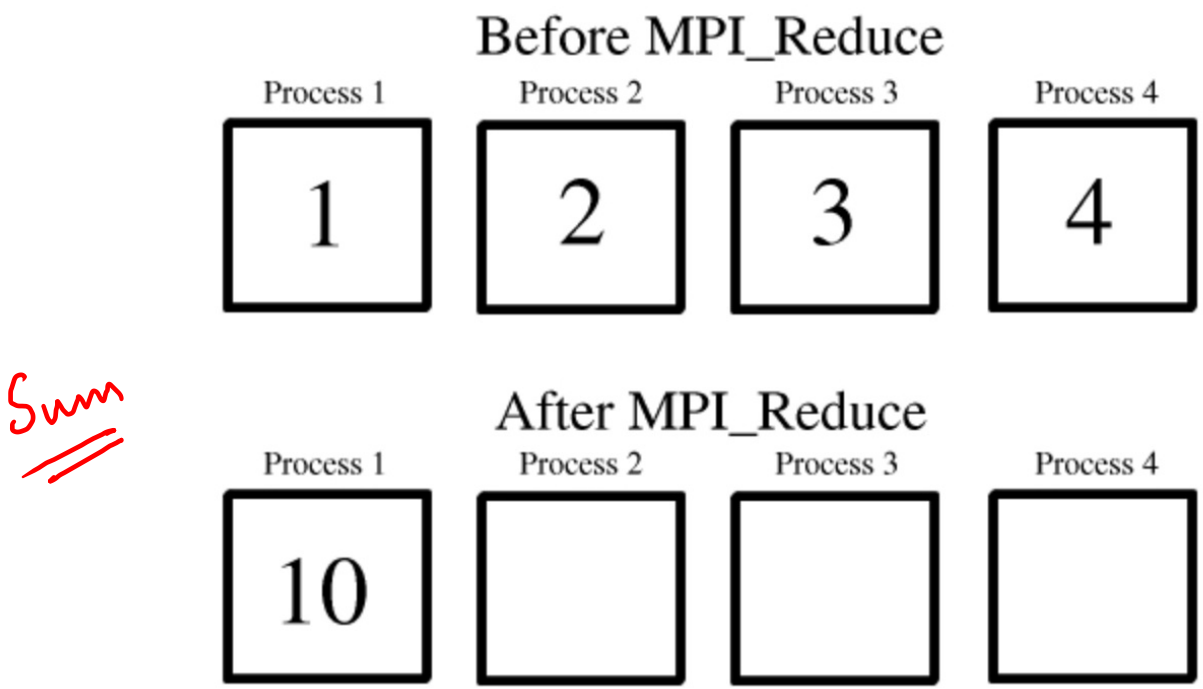
Parameter	Meaning of Parameter
sendbuf	starting address of send buffer (choice)
sendcount	number of elements in send buffer (integer)
sendtype	data type of send buffer elements (handle)
recvbuf	address of receive buffer (choice)
recvcount	number of elements received from any process (integer)
recvtype	data type of receive buffer elements (handle)
comm	communicator (handle)

MPI_Allgather gathers data from all tasks and distribute it to all.


Before MPI_Allgather
Process 1
Process 2
Process 3
Process 4
10
11
12
13
After MPI_Allgather
Process 1
Process 2
Process 3
Process 4
10
10
10
10
11
11
11
11
12
12
12
12
13
13
13
13

MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm);

Parameter	Meaning of Parameter
sendbuf	address of send buffer (choice)
recvbuf	address of receive buffer (choice, significant only at root)
count	number of elements in send buffer (integer)
datatype	data type of elements in send buffer (handle)
op	reduction operation (handle)
root	rank of root process (integer)
comm	communicator (handle)

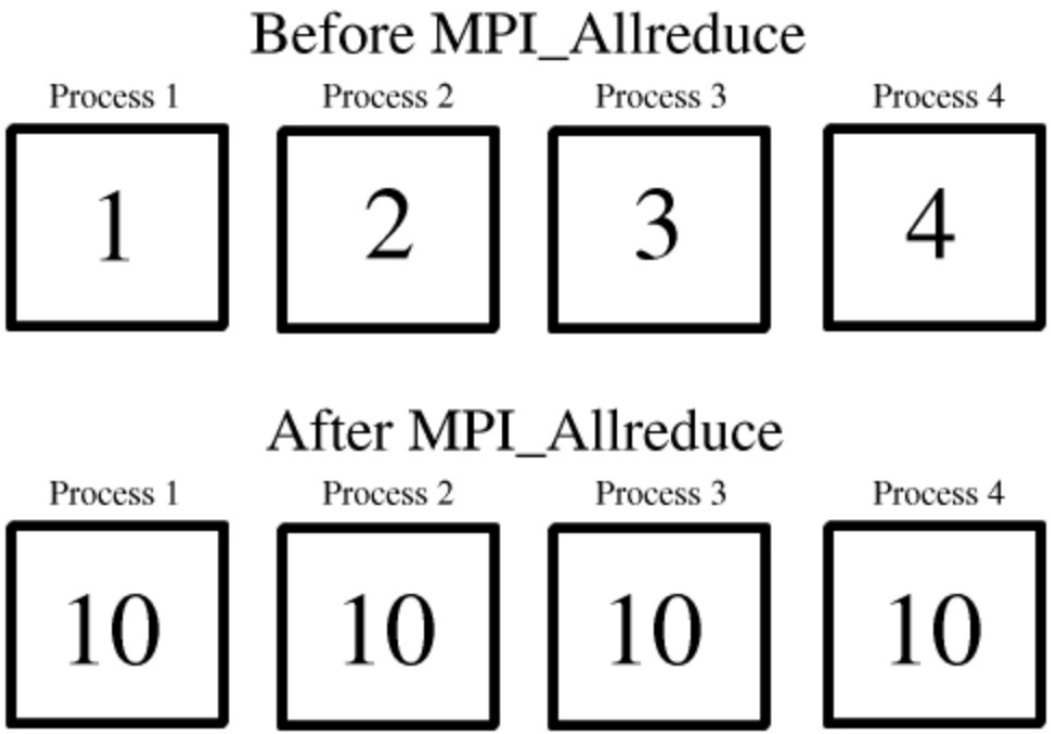

Before MPI_Reduce
Process 1
Process 2
Process 3
Process 4
1
2
3
4
After MPI_Reduce
Process 1
Process 2
Process 3
Process 4
10

Sum

MPI Reduction Operation	Meaning	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	Bitwise AND	integer, MPI_BYTE
MPI_LOR	Logical OR	integer
MPI_BOR	Bitwise OR	integer, MPI_BYTE
MPI_LXOR	Logical XOR	integer
MPI_BXOR	Bitwise XOR	integer, MPI_BYTE
MPI_MAXLOC	Maximum Value and Location	float, double and long double
MPI_MINLOC	Minimum Values and Location	float, double and long double

MPI_Allreduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm);

Parameter	**Meaning of Parameter**
sendbuf	address of send buffer (choice)
recvbuf	starting address of receive buffer (choice)
count	number of elements in send buffer (integer)
datatype	data type of elements in send buffer (handle)
op	operation (handle)
comm	communicator (handle)


Before MPI_Allreduce
Process 1
Process 2
Process 3
Process 4
1
2
3
4
After MPI_Allreduce
Process 1
Process 2
Process 3
Process 4
10
10
10
10

Sum