# *Module 4 : MPI Programming*

# *Collective Communication*

- Collective communication involves communication of data using all processes inside of a given communicator, the default communicator that contains all available processes is called MPI_COMM_WORLD.

- When a collective call is made it must be called by all processes inside of the communicatior.

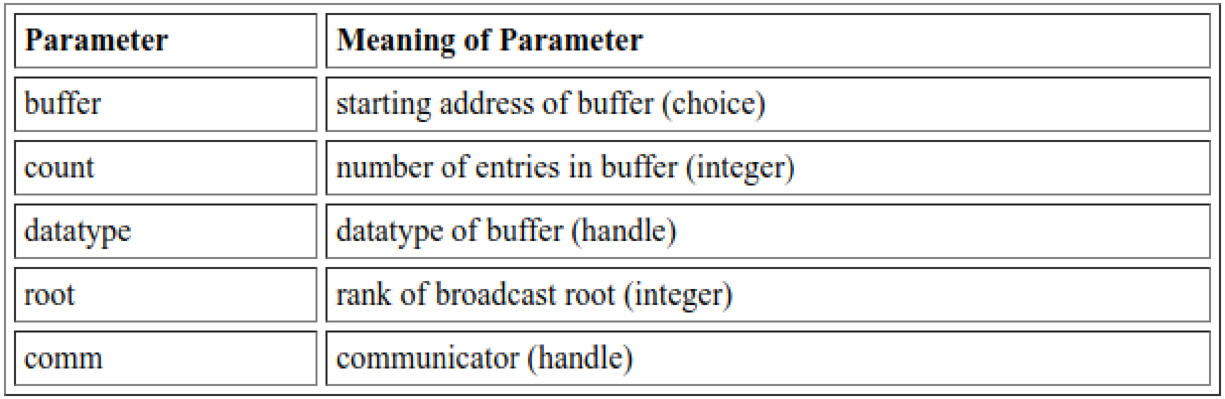
# *Types of collective communication*

Collective communication operations are made of the following types:

- Barrier Synchronization – Blocks until all processes have reached a synchronization point
- Data Movement (or Global Communication) – Broadcast, Scatters, Gather, All to All transmission of data across the communicator.
- Collective Operations (or Global Reduction) – One process from the communicator collects data from each process and performs an operation on that data to compute a result.Machine Learning

# *Barrier Synchronization*

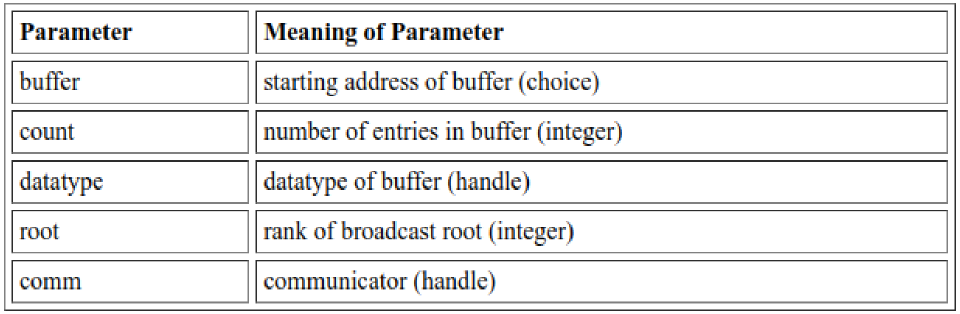
- **MPI_Barrier**
  - A barrier can be used to synchronize all processes in a communicator. Each process wait till all processes reach this point before proceeding further.
- **MPI_Bcast**
  - MPI_Bcast( void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm )



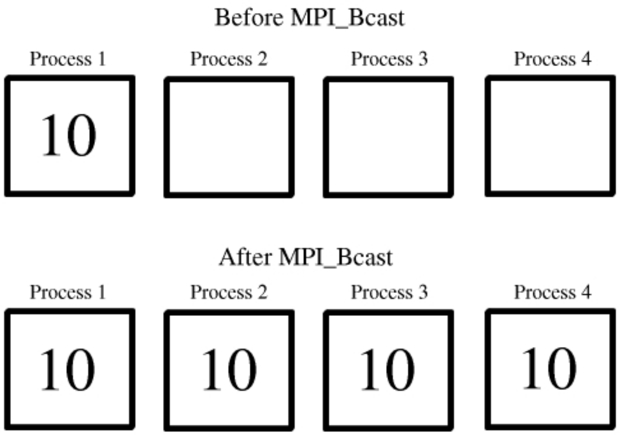
| **Parameter** | **Meaning of Parameter** |
|---|---|
| buffer | starting address of buffer (choice) |
| count | number of entries in buffer (integer) |
| datatype | datatype of buffer (handle) |
| root | rank of broadcast root (integer) |
| comm | communicator (handle) |

# Data Movement (or Global Communication)

- **MPI_Bcast**
  - MPI_Bcast( void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm )
  - MPI_Bcast broadcasts a message from the process with rank "root" to all other processes of the



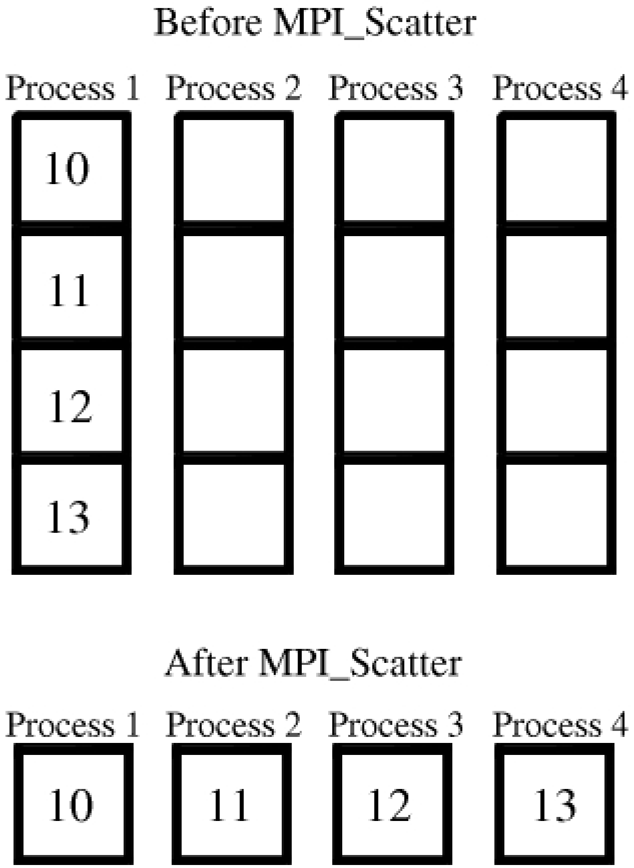
| Parameter | Meaning of Parameter |
|---|---|
| buffer | starting address of buffer (choice) |
| count | number of entries in buffer (integer) |
| datatype | datatype of buffer (handle) |
| root | rank of broadcast root (integer) |
| comm | communicator (handle) |



# *MPI_Scatter*

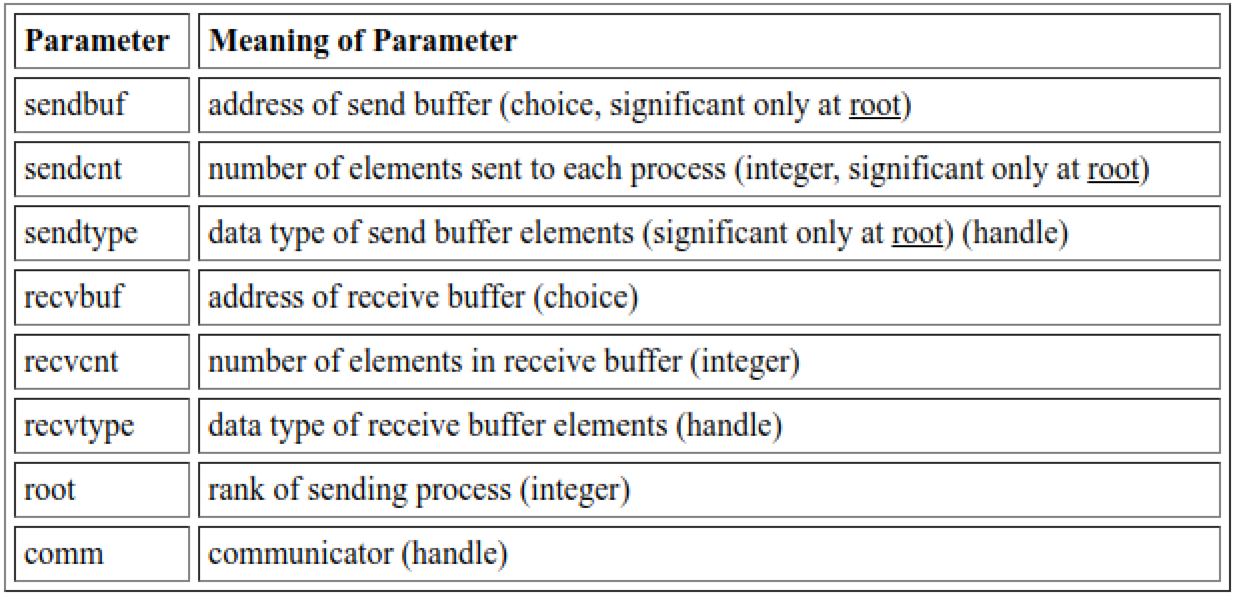
- MPI_Scatter sends data from one task to all other tasks in a group.

Given an array, divide it into equal contiguous parts and send to nodes, one part each. This is equivalent to n sends. The 0th process gets the first part, 1st processor the second part, and so on. Number of data elements to given to each node is specified in send count.



# *MPI_Scatter*

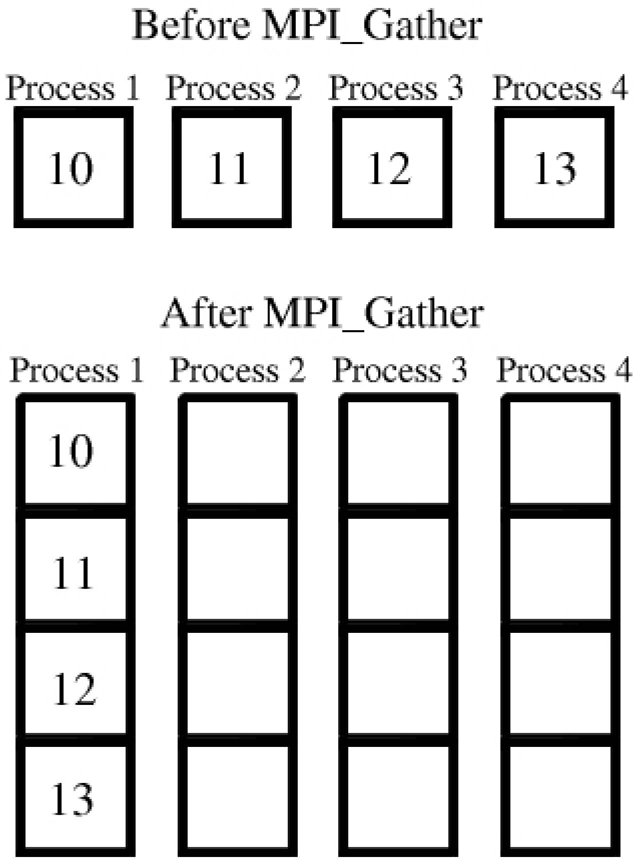
- MPI_Scatter( void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf, int recvcnt, MPI_Datatype recvtype, int root, MPI_Comm comm )



| 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) |

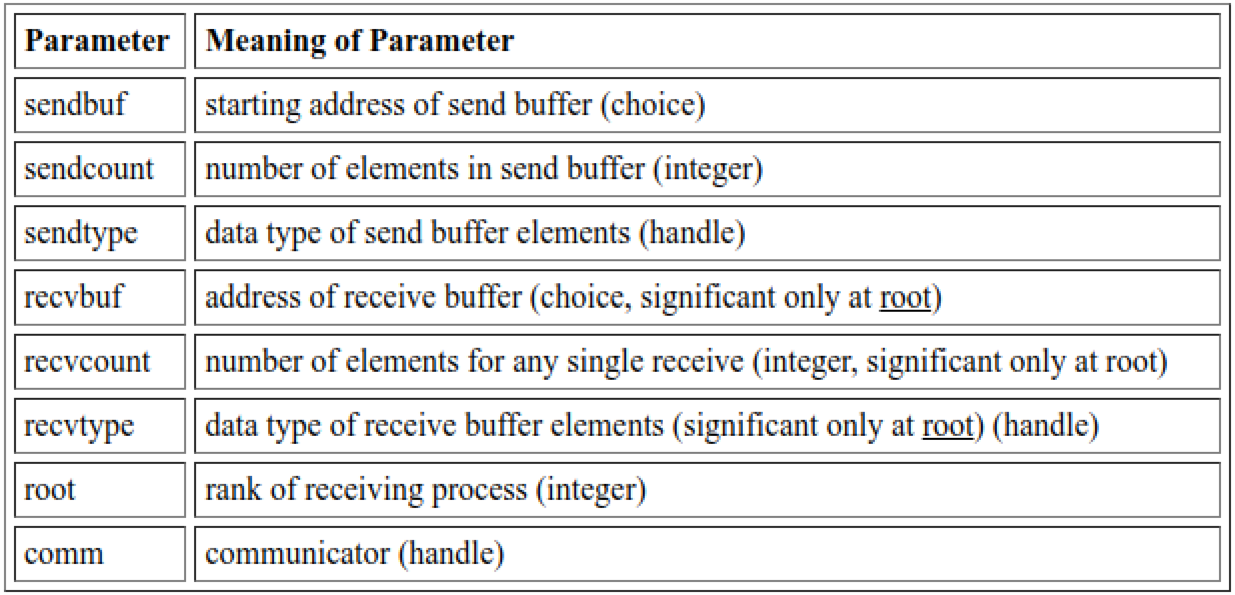
# *MPI_Gather*

- MPI_Gather gathers together values from a group of processes.



# MPI_Gather

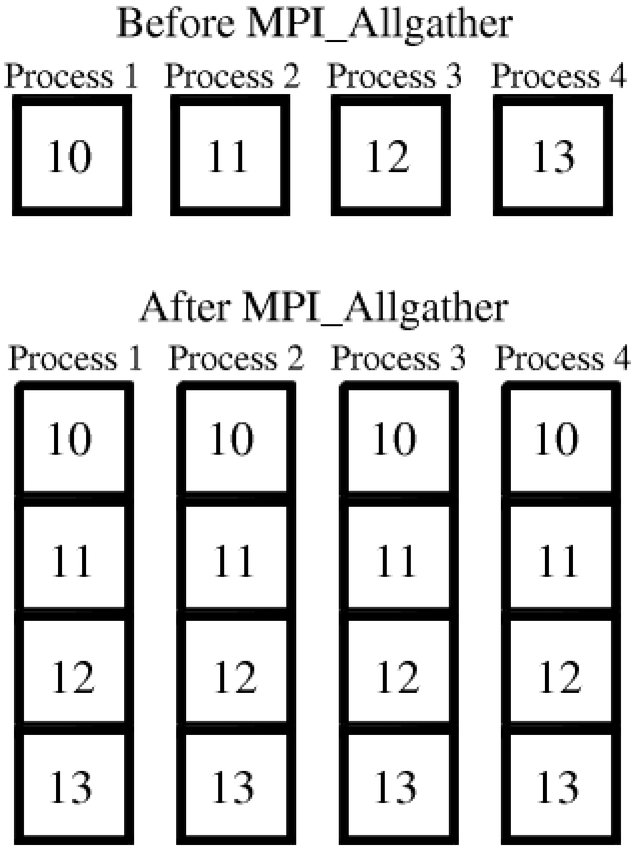
- MPI_Gather( void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm );



| 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_Allgather*

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



Before MPI_Allgather

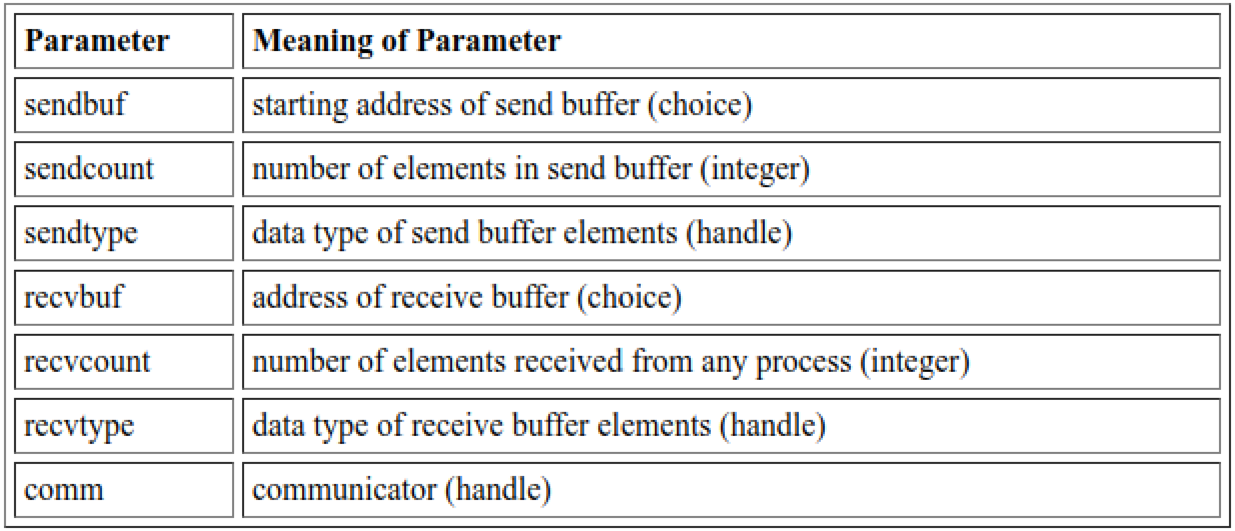
| Process 1 | Process 2 | Process 3 | Process 4 |
|---|---|---|---|
| 10 | 11 | 12 | 13 |

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_Allgather*

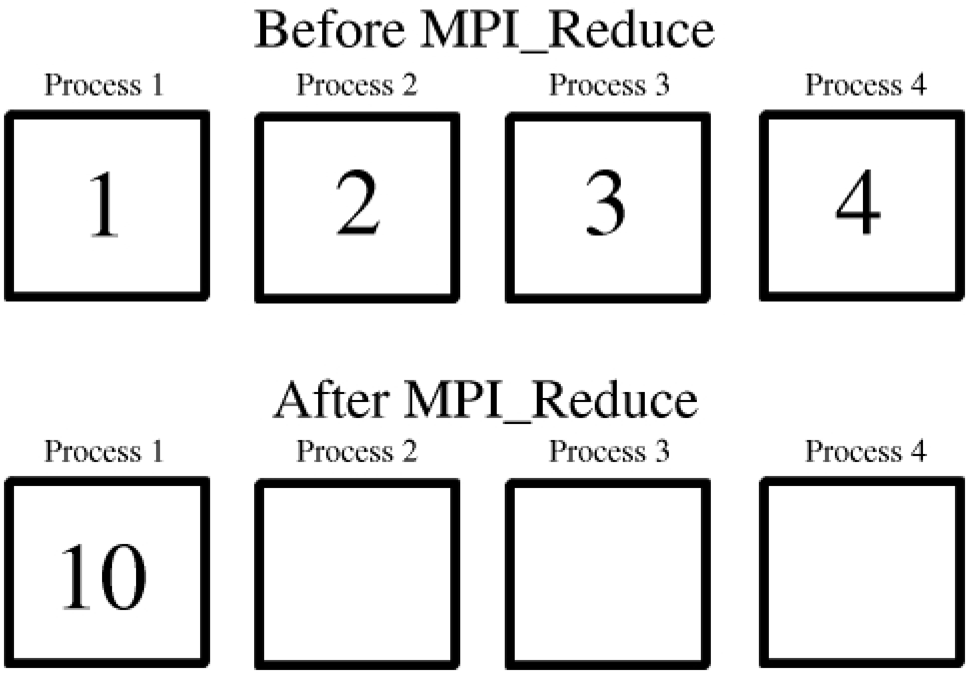
- MPI_Allgather( void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm );



| **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) |

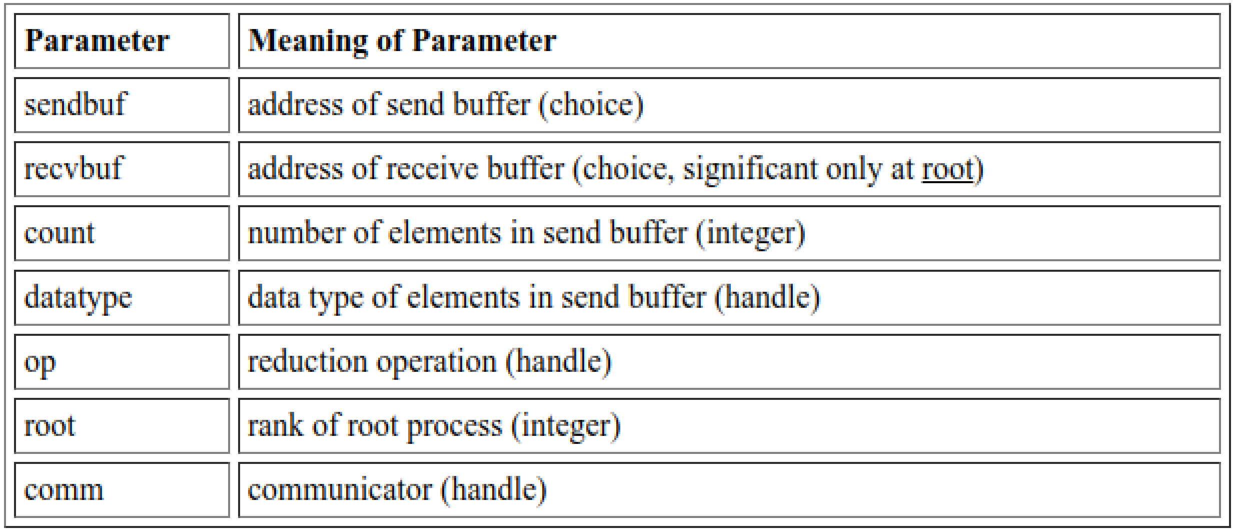
# *Collective Operations (or Global Reduction)*

- **MPI_Reduce -** MPI_Reduce reduces values on all processes to a single value.



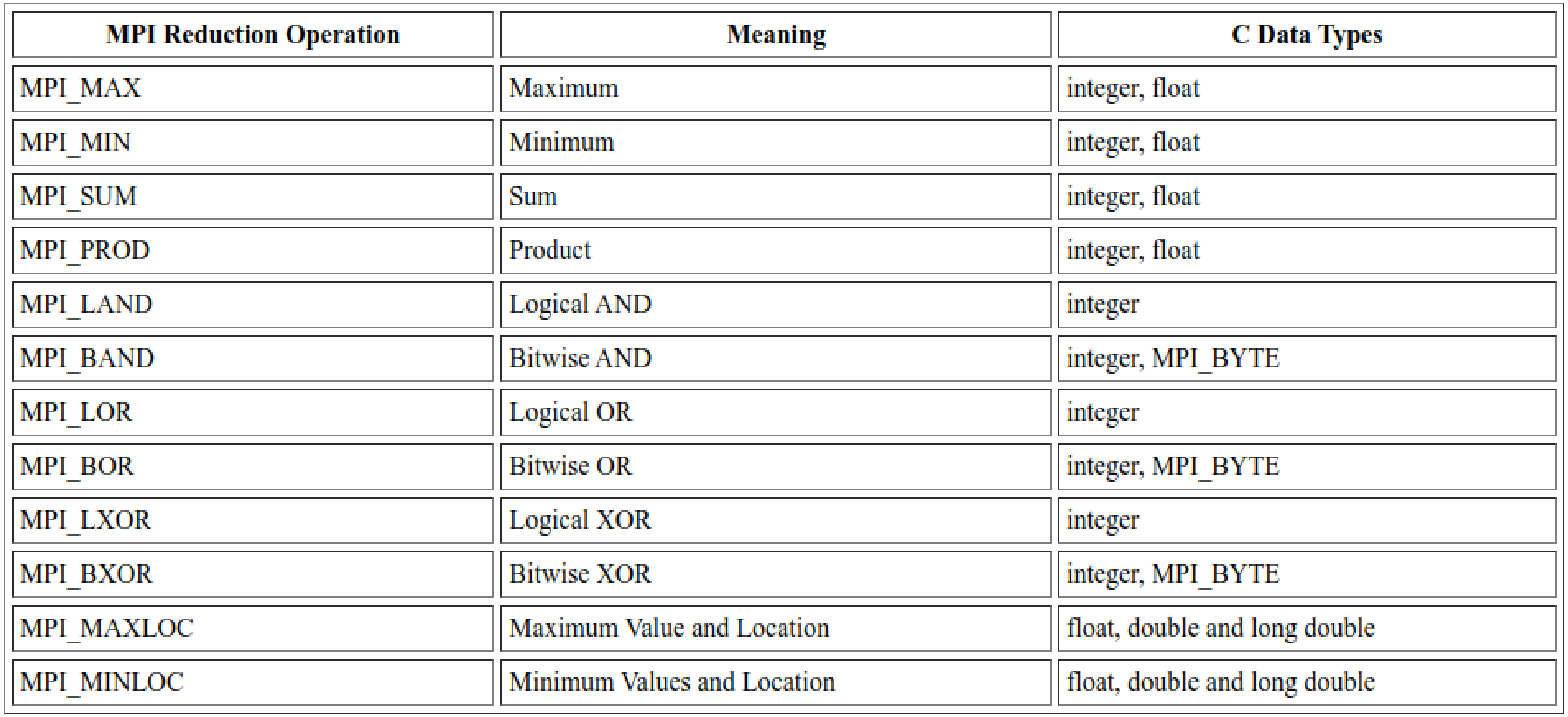
# MPI_Reduce

- 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) |

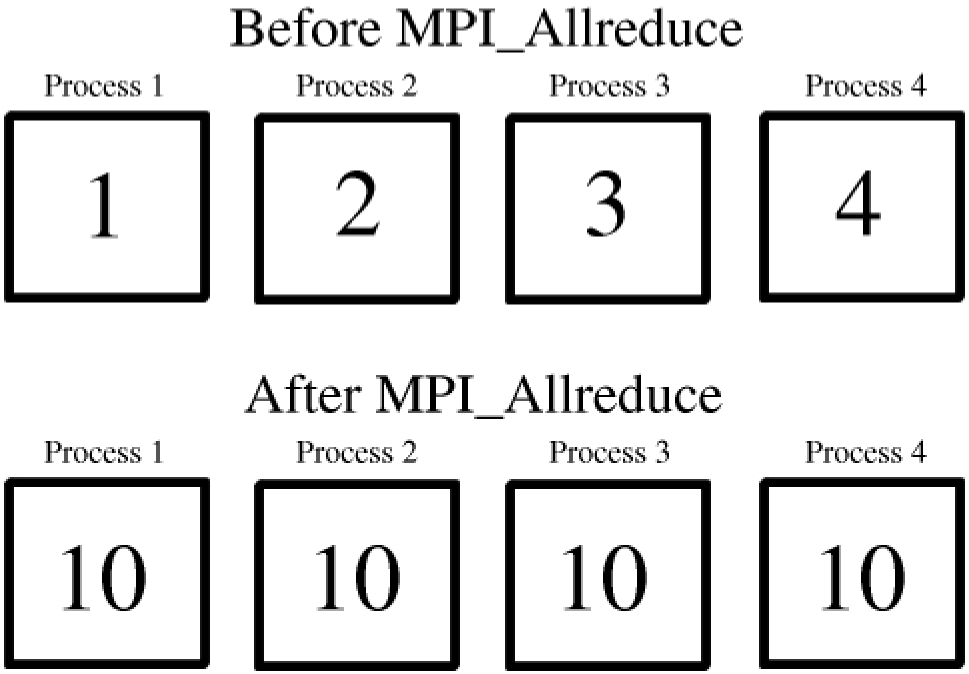
# MPI_Reduce - predefined reduction operations



| 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*

- MPI_Allreduce combines values from all processes and distribute the result back to all processes



Before MPI_Allreduce

Process 1 Process 2 Process 3 Process 4

1 2 3 4

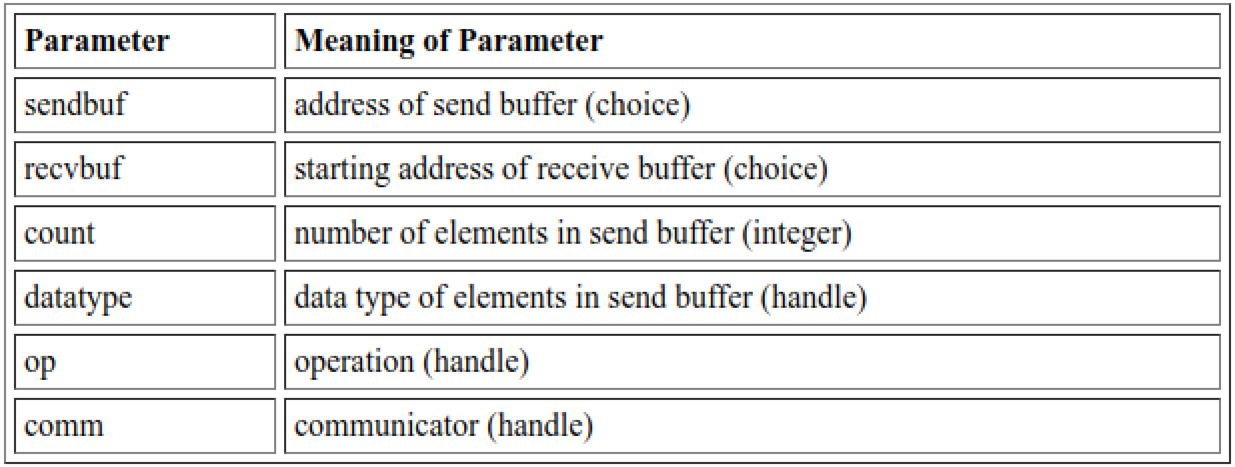
After MPI_Allreduce

Process 1 Process 2 Process 3 Process 4

10 10 10 10

# *MPI_Allreduce*

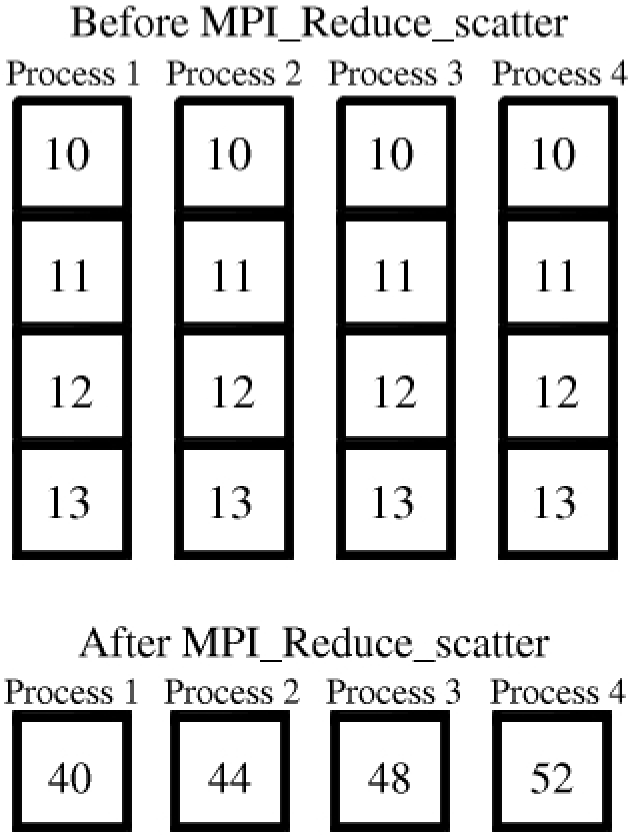
- MPI_Allreduce( void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm );



| **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) |

# *MPI_Reduce_scatter*

- MPI_Reduce_scatter combines values and scatters the results



Before MPI_Reduce_scatter

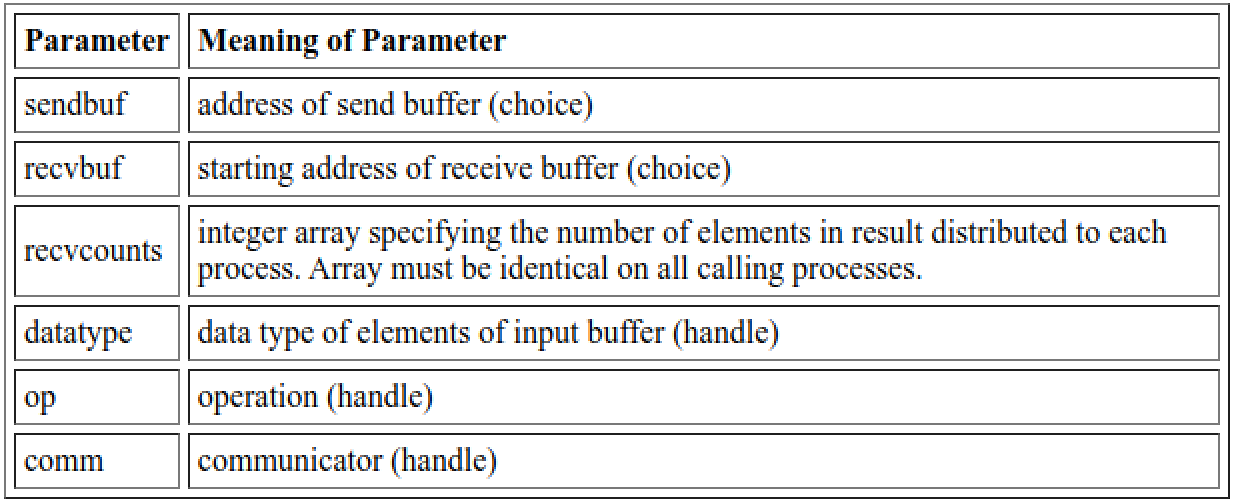
| Process 1 | Process 2 | Process 3 | Process 4 |
|---|---|---|---|
| 10 | 10 | 10 | 10 |
| 11 | 11 | 11 | 11 |
| 12 | 12 | 12 | 12 |
| 13 | 13 | 13 | 13 |

After MPI_Reduce_scatter

| Process 1 | Process 2 | Process 3 | Process 4 |
|---|---|---|---|
| 40 | 44 | 48 | 52 |

# *MPI_Reduce_scatter*

- MPI_Reduce_scatter( void *sendbuf, void *recvbuf, int *recvcounts, 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) |
| recvcounts | integer array specifying the number of elements in result distributed to each process. Array must be identical on all calling processes. |
| datatype | data type of elements of input buffer (handle) |
| op | operation (handle) |
| comm | communicator (handle) |