CEE 618 Scientific Parallel Computing (Lecture 5): Message-Passing Interface (MPI) advanced

Albert S. Kim

Department of Civil and Environmental Engineering
University of Hawai'i at Manoa
2540 Dole Street, Holmes 383, Honolulu, Hawaii 96822

Table of Contents

Outline

Review of MPI_BCAST

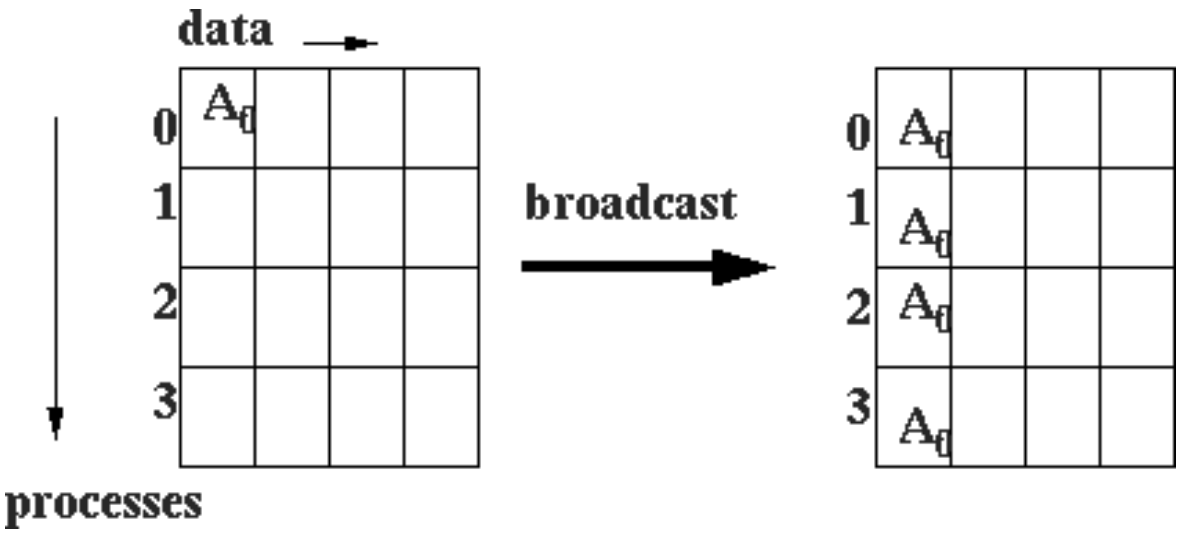


Figure: Schematic of broadcasting A_0 from process rank **0** to all processes (i.e., 0, 1, 2, and 3) **including itself**.

MPI_BCAST (Nbcast,1, MPI_INTEGER,**0**,MPI_COMM_WORLD,ierr)

- Two meanings of one MPI_BCAST call
  - If "myid" (rank) is "0", then send A_0 to everybody.
  - If "myid" (rank) is NOT "0", then receive A_0.

Review of MPI_REDUCE

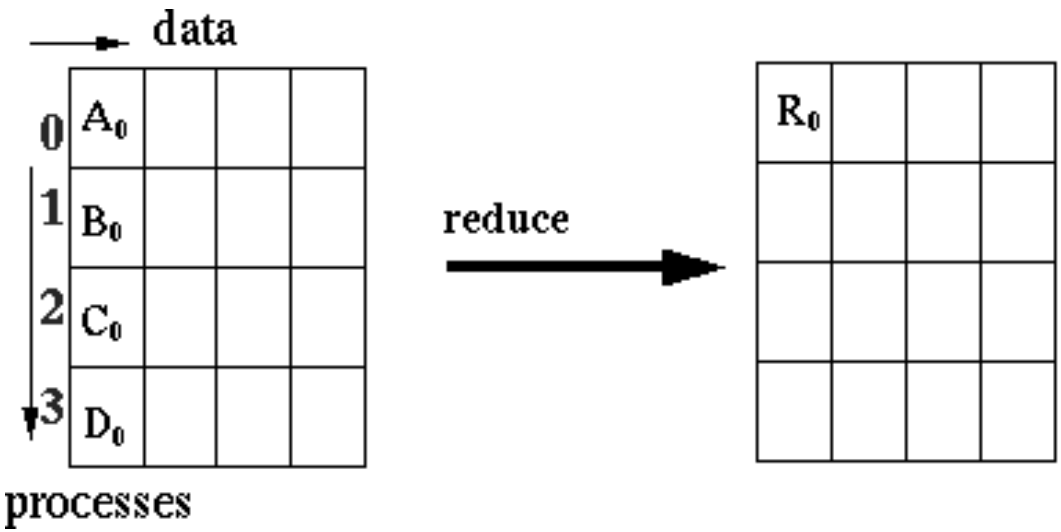


Figure: Schematic of reducing R_0 as $A_0 + B_0 + C_0 + D_0$ to process rank **0** if the operation is summation.

MPI_REDUCE (Ireduce, Nreduce, 1, MPI_INTEGER, **MPI_SUM**, **0**, MPI_COMM_WORLD, ierr)

- Other typical operations include **MPI_PROD**, **MPI_MAX**, and **MPI_MIN**.
- An **array** can be reduced.

Review of MPI_REDUCE

```
DOUBLE PRECISION, DIMENSION :: A(10), B(0:10)
...
REAL (KIND=16), DIMENSION :: C(100)
...
MPI_REDUCE (A, C, 2, MPI_DOUBLE_PRECISION,
MPI_SUM, 0, MPI_COMM_WORLD, ierr)
```

MPI_ALLREDUCE: Schematic

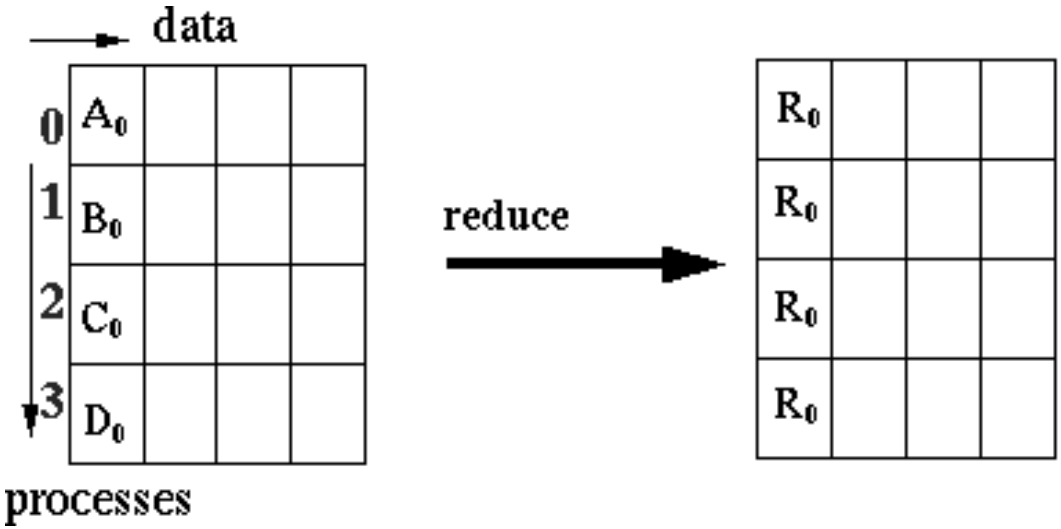


Figure: Schematic of reducing R_0 as $A_0 + B_0 + C_0 + D_0$ to **all process ranks** with MPI_SUM operation.

- MPI_ALLREDUCE is equivalent to MPI_REDUCE followed by MPI_BCAST. So, no need to specify "**who is reducing**".
- Example: MPI_ALLREDUCE (Ireduce, Nreduce, 1, MPI_INTEGER, MPI_SUM, ~~0,~~ MPI_COMM_WORLD, ierr)
- Compare: MPI_REDUCE (Ireduce, Nreduce, 1, MPI_INTEGER, MPI_SUM, 0, MPI_COMM_WORLD, ierr)

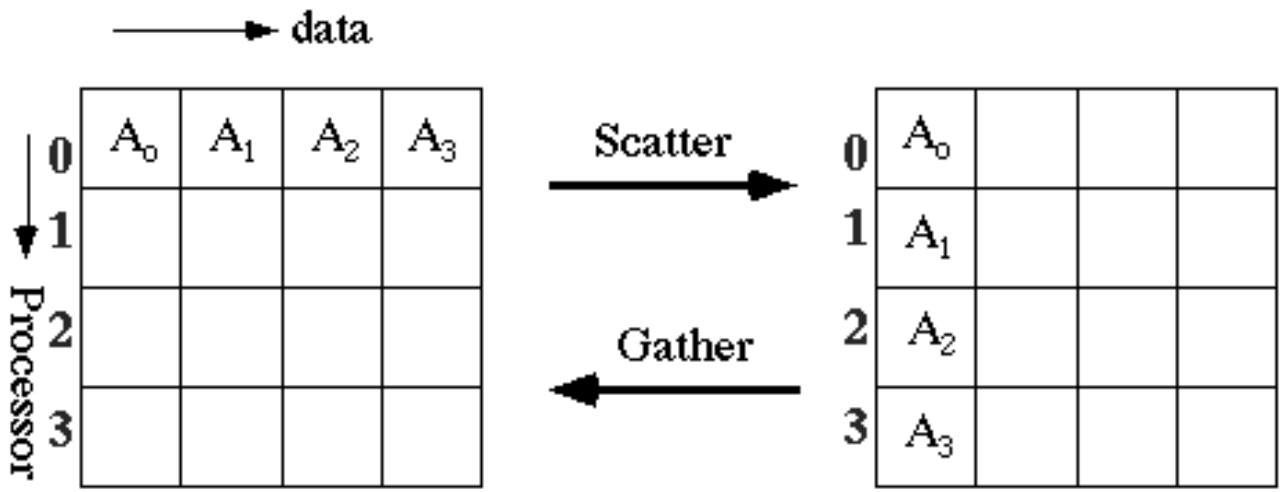
MPI_ALLREDUCE: Usage

MPI_ALLREDUCE (Ireduce, Nreduce, 1, MPI_INTEGER, MPI_SUM, ~~0,~~ MPI_COMM_WORLD, ierr)

- MPI_ALLREDUCE (sendbuf, recvbuf, count, datatype, op, ~~root,~~ comm, ierr) reduces values on all processes to a **single** value (if **count=1**).
  1. sendbuf - address of send buffer (choice)
  2. recvbuf - address of receive buffer (choice)
  3. count - number of elements in send buffer (integer)
  4. datatype - data type of elements of send buffer (handle)
  5. op - reduce operation (handle)
  6. comm - communicator (handle)

MPI_ALLREDUCE reduces “sendbuf” of each processes as “recvbuf” of all the processes with “op” operation. No need to specify “**the root**”.

MPI_GATHER: Schematic



- C/C++ format: A_0, A_1, A_2, and A_3.
- FORTRAN format: A_1, A_2, A_3 and A_4.

MPI_GATHER: Usage

MPI_GATHER (sendbuf, sendcnt, sendtype, recvbuf, recvcount, recvtype, **root**, comm, ierr)

- sendbuf - starting address of send buffer (choice)
- sendcount - number of elements in send buffer (integer)
- sendtype - data type of send buffer elements (handle)
- recvcount - number of elements received from any process (integer)
- recvtype - data type of receive buffer elements (handle)
- **root** - rank of receiving process (integer)
- comm - communicator (handle)

MPI_GATHER: Example Code

```
program gather
implicit none
include 'mpif.h'
integer numprocs, rank, ierr, rc
integer a(54), b(54),  Nscat, i

call MPI_INIT(ierr)
if (ierr .ne. MPI_SUCCESS) then
   print *,'Error starting MPI program. Terminating.'
   call MPI_ABORT(MPI_COMM_WORLD, rc, ierr)
end if

call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, numprocs, ierr)

Nscat = -10

print *, 'I am ',rank,' of ',numprocs, 'and Nbcast =', Nscat

call MPI_BARRIER(MPI_COMM_WORLD,ierr)
```

MPI_GATHER: Example Code (cont'd)

```
if (rank.eq.0) then
   do i = 1, numprocs
      a(i) = i*i
   end do
end if

call MPI_SCATTER (a,1,MPI_INTEGER,Nscat,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)

call MPI_GATHER  (Nscat,1,MPI_INTEGER,b,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)

print *, 'I am ',rank,' of ',numprocs, 'and Nbcast =', Nscat

if (rank.eq.0) then
   do i = 1, numprocs
      write(*,"('b(',i1,')= ',i6)") i, b(i)
   end do
end if
call MPI_FINALIZE(ierr)

end
```

MPI_GATHER: Procedure

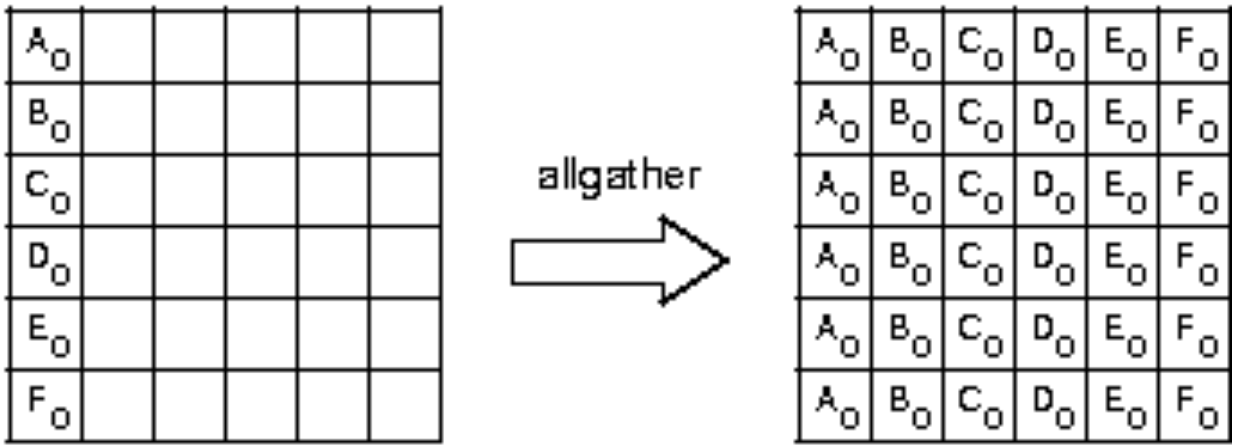
1. All the processes have $N_{\mathrm{scat}} = -10$.
2. Process rank 0 calculates
   $a(1) = 1$, $a(2) = 4$, $a(3) = 9$, and $a(4) = 16$.
3. Process rank 0 scatters a to all the processes as N_{scat}.
   MPI_SCATTER (**a**, 1, MPI_INTEGER, **Nscat**, 1, MPI_INTEGER, **0**, MPI_COMM_WORLD, ierr)
   - Process rank 0 sends $a(1)$ to process 0 as rank 0's $N_{\mathrm{scat}} = 1$.
   - Process rank 0 sends $a(2)$ to process 1 as rank 1's $N_{\mathrm{scat}} = 4$.
   - Process rank 0 sends $a(3)$ to process 2 as rank 2's $N_{\mathrm{scat}} = 9$.
   - Process rank 0 sends $a(4)$ to process 3 as rank 3's $N_{\mathrm{scat}} = 16$.
4. Process rank 1-3 do not have specifically assigned $a(i)$.
5. Process rank 0 gathers N_{scat} of all the processes as $b(i)$.
   MPI_GATHER (**Nscat**, 1, MPI_INTEGER, **b**, 1, MPI_INTEGER, **0**, MPI_COMM_WORLD, ierr)
   - Process rank 0 gathers rank 0's $N_{\mathrm{scat}} = 1$ as rank 0's $b(1)$.
   - Process rank 0 gathers rank 1's $N_{\mathrm{scat}} = 4$ as rank 0's $b(2)$.
   - Process rank 0 gathers rank 2's $N_{\mathrm{scat}} = 9$ as rank 0's $b(3)$.
   - Process rank 0 gathers rank 3's $N_{\mathrm{scat}} = 16$ as rank 0's $b(4)$.

MPI_GATHER: Result using 4 processors

```
 I am           0  of        4 and Nbcast =        -10
 I am           2  of        4 and Nbcast =        -10
 I am           1  of        4 and Nbcast =        -10
 I am           3  of        4 and Nbcast =        -10
 I am           3  of        4 and Nbcast =         16
 I am           2  of        4 and Nbcast =          9
 I am           1  of        4 and Nbcast =          4
 I am           0  of        4 and Nbcast =          1
b(1)=           1
b(2)=           4
b(3)=           9
b(4)=          16
```

Note that $a(1)$ to $a(4)$ are used out of $a(54)$ where '54' is the dimension (or the number of elements) of array a and '4' is the number of processors used for this MPI run.

MPI_ALLGATHER: Schematic



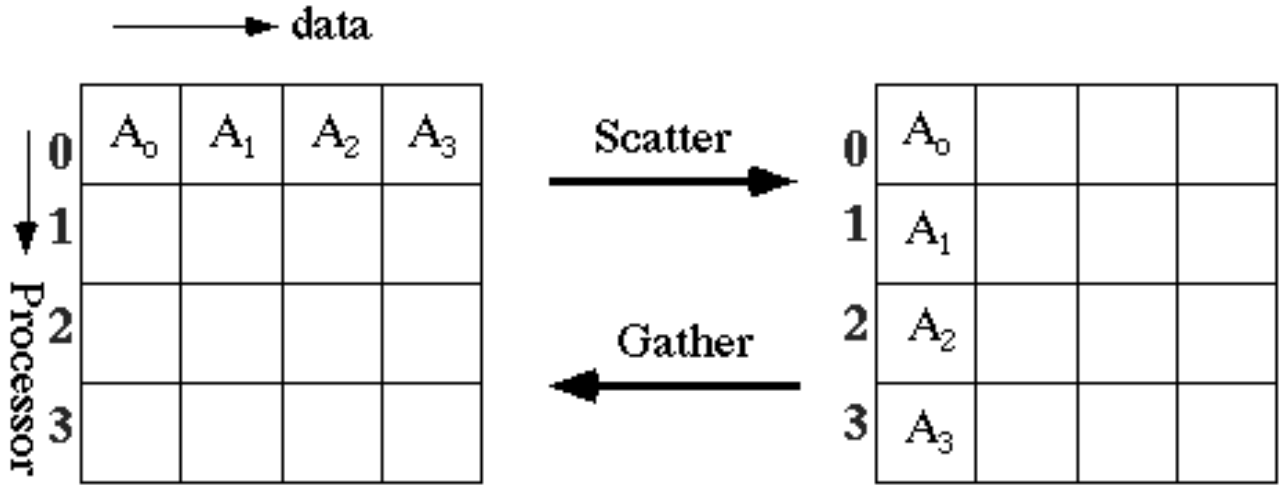
- C/C++ format: A_0, B_0, C_0, D_0, E_0, F_0
- FORTRAN format: A_1, B_1, C_1, D_1, E_1, F_1

MPI_ALLGATHER: Usage

MPI_**ALL**GATHER
(sendbuf,sendcount,sendtype,recvbuf,recvcount,recvtype,
comm,info)
MPI_GATHER
(sendbuf,sendcount,sendtype,recvbuf,recvcount,recvtype, **root**,
comm,ierr)

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

MPI_SCATTER: Schematic



- C/C++ format: A_0, A_1, A_2, and A_3.
- FORTRAN format[1]: A_1, A_2, A_3 and A_4.

[1] If you want to use 0 index, then you can declare **a(0:3)** instead of a(4) in fortran codes.

MPI_SCATTER: Example Code

```
program scatter
implicit none
include 'mpif.h'
integer numprocs, rank, ierr, rc
integer a(54), Nscat, i

call MPI_INIT(ierr)
if (ierr .ne. MPI_SUCCESS) then
   print *,'Error starting MPI program. Terminating.'
   call MPI_ABORT(MPI_COMM_WORLD, rc, ierr)
end if

call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, numprocs, ierr)

Nscat = -10

print *, 'I am ',rank,' of ',numprocs, 'and Nbcast =', Nscat

call MPI_BARRIER(MPI_COMM_WORLD,ierr)
```

MPI_SCATTER: Example Code (cont'd)

```
if (rank.eq.0) then
   do i = 1, numprocs*2
      a(i) = i*i
   end do
end if

call MPI_SCATTER (a,1, MPI_INTEGER,Nscat,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)

print *, 'I am ',rank,' of ',numprocs, 'and Nbcast =', Nscat


call MPI_FINALIZE(ierr)

end
```

MPI_SCATTER: Procedure

1. All the processes have $N_{\mathrm{scat}} = -10$.
2. Process rank 0 calculates $a(1) = 1$, $a(2) = 4$, $a(3) = 9$, and $a(4) = 16$.
3. Process rank 0 scatters a to all the processes as their N_{scat}.
   MPI_SCATTER (**a**, 1, MPI_INTEGER, **Nscat**, 1, MPI_INTEGER, **0**, MPI_COMM_WORLD, ierr)
   - Process rank 0 send $a(1)$ to process 0 as rank 0's $N_{\mathrm{scat}} = 1$.
   - Process rank 0 send $a(2)$ to process 1 as rank 1's $N_{\mathrm{scat}} = 4$.
   - Process rank 0 send $a(3)$ to process 2 as rank 2's $N_{\mathrm{scat}} = 9$.
   - Process rank 0 send $a(4)$ to process 3 as rank 3's $N_{\mathrm{scat}} = 16$.
4. Process rank 1-3 do not have specifically assigned $a(i)$, which are at least different from what rank 0 has.

MPI_SCATTER: Result

```
I am      1  of      4 and Nbcast =      -10
I am      2  of      4 and Nbcast =      -10
I am      3  of      4 and Nbcast =      -10
I am      0  of      4 and Nbcast =      -10
I am      0  of      4 and Nbcast =        1
I am      1  of      4 and Nbcast =        4
I am      2  of      4 and Nbcast =        9
I am      3  of      4 and Nbcast =       16
```

- MPI does not have MPI_ALLSCATTER routine. Why?
  (ANS.) **MPI_BCAST**

Outline

MPI_SEND & MPI_RECV: Example Code

```
program sendrecv
implicit none
include 'mpif.h'
integer :: numprocs, rank, ierr, rc
integer :: N, irank, mytag, stat(MPI_STATUS_SIZE)
character (len=1) :: gotit

call MPI_INIT(ierr)
if (ierr .ne. MPI_SUCCESS) then
   print *,'Error starting MPI program. Terminating.'
   call MPI_ABORT(MPI_COMM_WORLD, rc, ierr)
end if

call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, numprocs, ierr)

N = -10
print *, 'I am ',rank,' of ',numprocs, 'and N =', N
call MPI_BARRIER(MPI_COMM_WORLD,ierr)
```

MPI_SEND & MPI_RECV: Example Code (cont'd)

```
!=============================================
if (rank.eq.0)  then
  N = 10
  do irank = 1, numprocs-1
    mytag = 1000 + irank
    call MPI_SEND &
       ( N, 1, MPI_INTEGER, irank, mytag, MPI_COMM_WORLD, ierr)
  end do
  mytag = 1000 + rank

  write(*,"('I am ',i2,' of ',i2,'. I changed N from  -10 to ' &
        ,i3,' and broadcasted it.')") rank, numprocs, N
  call MPI_BARRIER(MPI_COMM_WORLD,ierr)

  gotit = 'N'
  call MPI_BARRIER(MPI_COMM_WORLD,ierr)
  do irank = 1, numprocs-1
    call MPI_RECV ( gotit, 1, MPI_CHARACTER, MPI_ANY_SOURCE, &
                    MPI_ANY_TAG,  MPI_COMM_WORLD, stat, ierr)
    write(*,"('I am ',i2,' of ',i2,' and my gotit =',a2,' &
              from ',i2,' with mytag =',i6)") &
          rank, numprocs, gotit, stat(MPI_SOURCE), stat(MPI_TAG)
  end do


!=============================================
```

```
!=============================================
else


  call MPI_RECV ( N, 1, MPI_INTEGER, 0, MPI_ANY_TAG,     &
           MPI_COMM_WORLD, stat, ierr)
  mytag = stat(MPI_TAG)



  call MPI_BARRIER(MPI_COMM_WORLD,ierr)
  write(*,                                                  &
      "('I am ',i2,' of ',i2,' and my N =',i2,' with mytag=',i6)") &
       rank, numprocs, N, mytag

  gotit = 'Y'
  call MPI_SEND ( gotit, 1, MPI_CHARACTER, 0, mytag, &
                  MPI_COMM_WORLD, ierr)

  call MPI_BARRIER(MPI_COMM_WORLD,ierr)



end if

call MPI_FINALIZE(ierr)
end
```

MPI_SEND & MPI_RECV: Result

```
 I am            2  of            4 and N =         -10
 I am            1  of            4 and N =         -10
 I am            0  of            4 and N =         -10
 I am            3  of            4 and N =         -10
I am  0 of  4. I changed N from -10 to  10 and broadcasted it.
I am  1 of  4 and my N =10 with mytag=  1001
I am  3 of  4 and my N =10 with mytag=  1003
I am  2 of  4 and my N =10 with mytag=  1002
I am  0 of  4 and my gotit = Y  from  1 with mytag =  1001
I am  0 of  4 and my gotit = Y  from  2 with mytag =  1002
I am  0 of  4 and my gotit = Y  from  3 with mytag =  1003
```

The communication outcome is exactly same as what obtained by using MPI_BCAST, i.e.,

MPI_BCAST (**N**, 1, MPI_INTEGER, **0**, MPI_COMM_WORLD, ierr)

with some additional information.

MPI_SEND: Usage

- MPI_SEND (sendbuf, count, datatype, dest, tag, comm, ierr)
  send data from one rank to a specific rank in a group.
  1. sendbuf - address of send buffer (choice)
  2. count - number of elements in send buffer (integer)
  3. datatype - data type of elements of send buffer (handle)
  4. dest - rank of destination (integer)
  5. **tag** - message tag (integer)
  6. comm - communicator (handle)

MPI_RECV: Usage

- MPI_SEND (*sendbuf*, count, datatype, *dest*, tag, comm, ierr) send data from one rank to a specific rank in a group.

- MPI_RECV (*recvbuf*, count, datatype, *source*, tag, comm, **status**, ierr) receives data from a specific rank in a group.
  1. recvbuf - address of receive buffer (choice)
  2. count - number of elements in send buffer (integer)
  3. datatype - data type of elements of send buffer (handle)
  4. source - rank of source (integer)
  5. tag - message tag (integer)
  6. comm - communicator (handle)
  7. **status** - status object (Status)

How they work

- call MPI_SEND (**N**, 1, MPI_INTEGER, **irank**, mytag, MPI_COMM_WORLD, ierr)
  1. Process rank 0 sends *N* to process **irank**= 1, 2, ..., and numprocs-1.
  2. mytag = 1000 + irank
- call MPI_RECV (**N**, 1, MPI_INTEGER, **0**, **MPI_ANY_TAG**, MPI_COMM_WORLD, **stat**, ierr)
  1. Process ranks *irank= 1, 2, ..., and numprocs-1* receive *N* from process rank **0**.
  2. MPI_ANY_TAG indicates **any tag** is acceptable.
  3. stat has information of **senders** and **tag values**.

Tag and Status

- For Sender
  1. A specific *tag* provides additional information of sent data, e.g., a giver ID of a birthday present.
- For Receiver
  1. A receiver can take data of a specific tag value or take any tag by using MPI_ANY_TAG.
  2. Tag value and sender ID are stored in stat.
     - sender = stat(**MPI_SOURCE**)
     - tag = stat(**MPI_TAG**)
- stat (MPI_STATUS_SIZE) has 5 components.
  - MPI_STATUS_SIZE=**5**
  - MPI_SOURCE =**1**
  - MPI_TAG = **2**

Outline

programming tips

- Prepare three source files for main, master, and worker: main.f, master.f, and worker.f.
- Have main.f only for basic settings of the program.
- For master part, write in the main.f file, **include 'main.f'**
- For worker part, write in the main.f file, **include 'worker.f'**

```
if(myid.eq.0) then
  include "master.f"
else
  include "worker.f"
endif
```

vi editor tips

- After you open any source code, try
  1. “Esc” and “:vsp” or
  2. “Esc” and “:sp”
- To switch, “Ctrl+Shift+ww”
- To close one, “:wq”