Parallel Computing - MPI

Message Passing Interface

By : Om Jadhav
HPC - Tech, CDAC Pune

MPI - Message Passing Interface

MPI is built on ‘Routines’

The basic MPI Routines :-

- MPI_Init () ;
- MPI_Comm_rank () ;
- MPI_Comm_size () ;
- MPI_Send () ;
- MPI_Recv () ;
- MPI_Finalize () ;

- - - -- - - - - - - - -

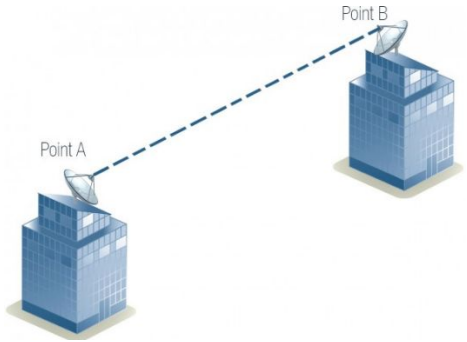
MPI - Communication

MPI - Communication

Point to Point Comm^{n}

MPI - Communication

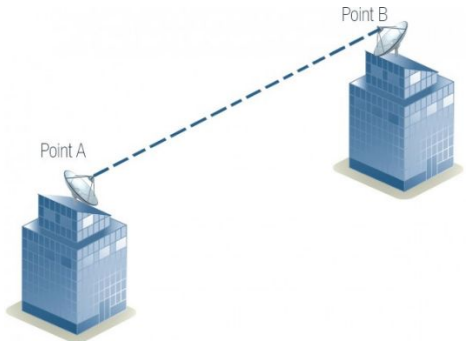
Point to Point Comm^{n}



MPI - Communication

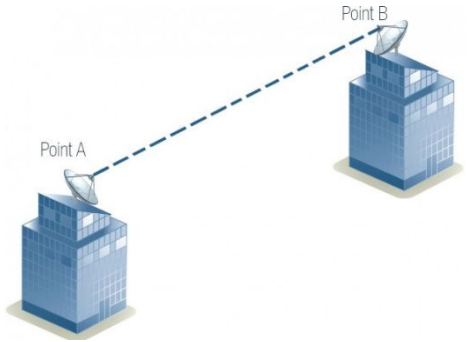
Point to Point Comm^{n}

Collective Comm^{n}

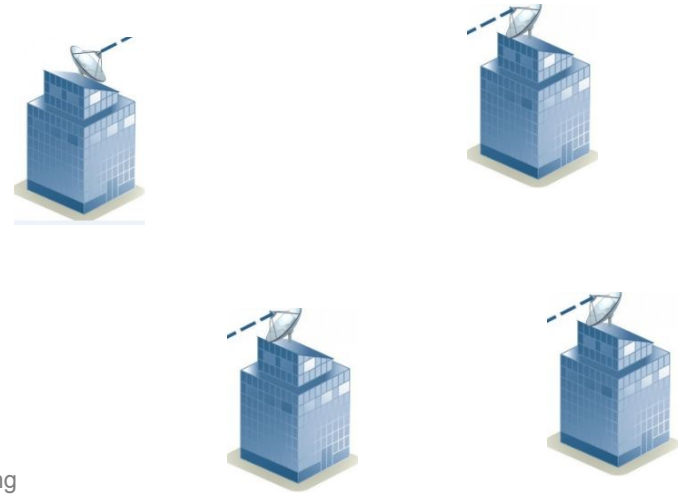


MPI - Communication

Point to Point Comm^{n}

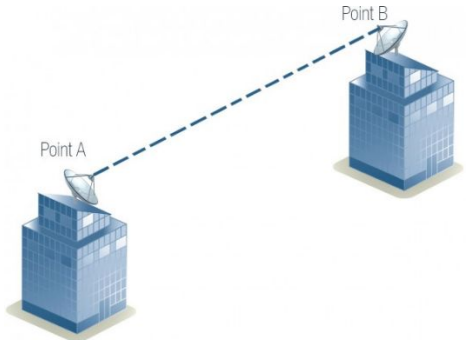


Collective Comm^{n}

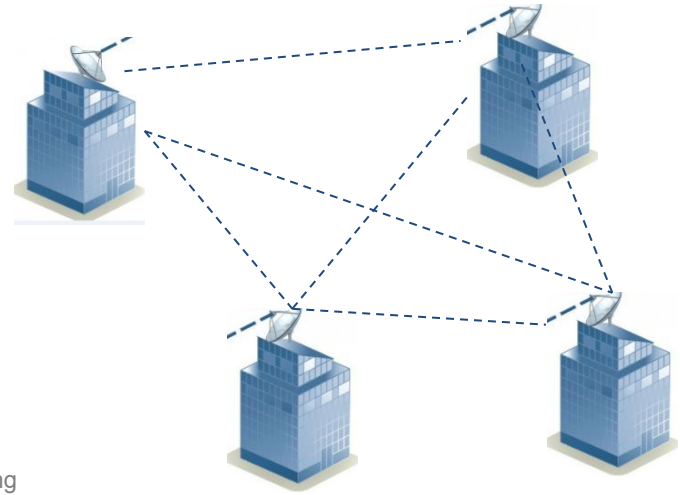


MPI - Communication

Point to Point Comm^{n}

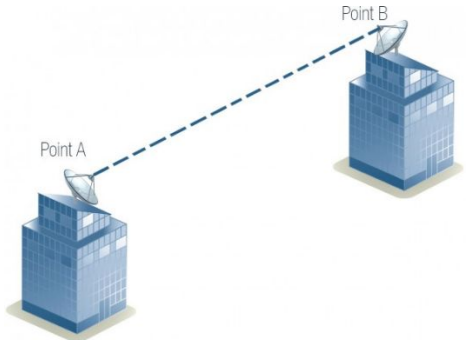


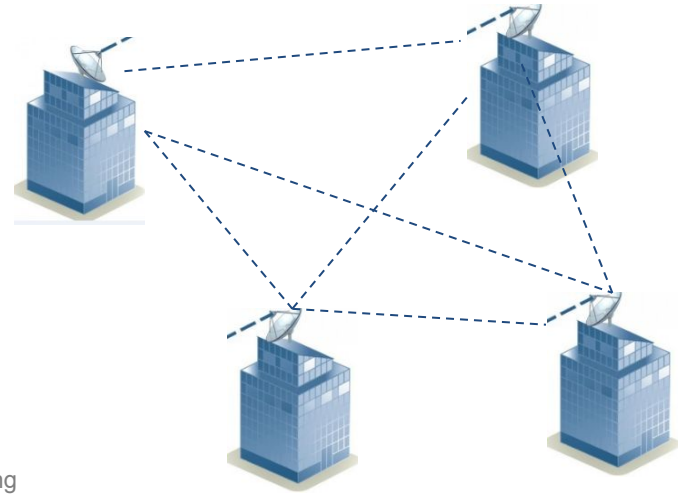
Collective Comm^{n}



MPI - Communication

Point to Point Comm^{n} → **Collective Comm^{n}**





MPI - Collective Communication

- Collective communication must involve all processes in the scope of a communicator.
- Involve coordinated communication within a group of processes identified by an MPI communicator.

Types of Collective Operations

- Synchronization - Processes wait until all members of the group have reached the synchronization point.
- Data Movement - broadcast, scatter/gather, all to all
- Collective Computation (reductions) - one member of the group collects data from the other members and performs an operation (min,max, add, multiply, etc.) on that data.

Basic Collective Communication Routines

- MPI_Bcast( ) - Broadcast (one to all)
- MPI_Scatter( ) - Scatter (one to all)
- MPI_Gather( ) - Gather (all to one)
- MPI_Reduce( ) - Reduce (all to one)
- MPI_Allgather( ) - (all to all)
- MPI_Allreduce( ) - (all to all)

MPI - Broadcast

Syntax :

MPI_Bcast (void* data , Int count , MPI_Datatype datatype , Int source_process , MPI_Comm comm) ;

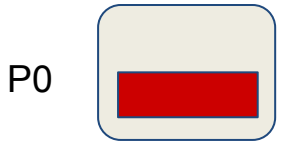
- One process sends the same data to all processes in a communicator.

MPI - Broadcast

Syntax :

MPI_Bcast (void* data , Int count , MPI_Datatype datatype , Int source_process , MPI_Comm comm) ;

- One process sends the same data to all processes in a communicator.

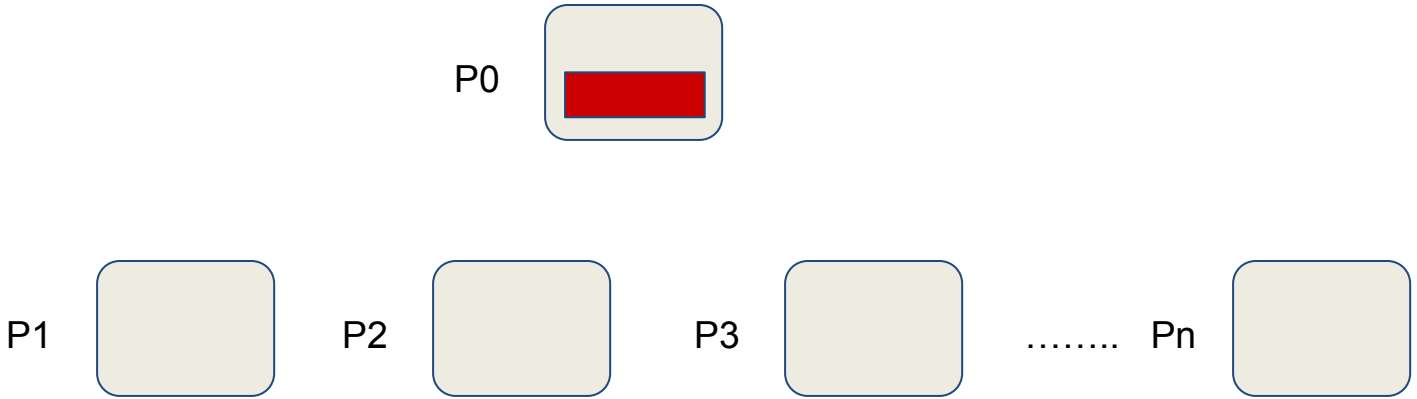


MPI - Broadcast

Syntax :

MPI_Bcast (void* data , Int count , MPI_Datatype datatype , Int source_process , MPI_Comm comm) ;

- One process sends the same data to all processes in a communicator.

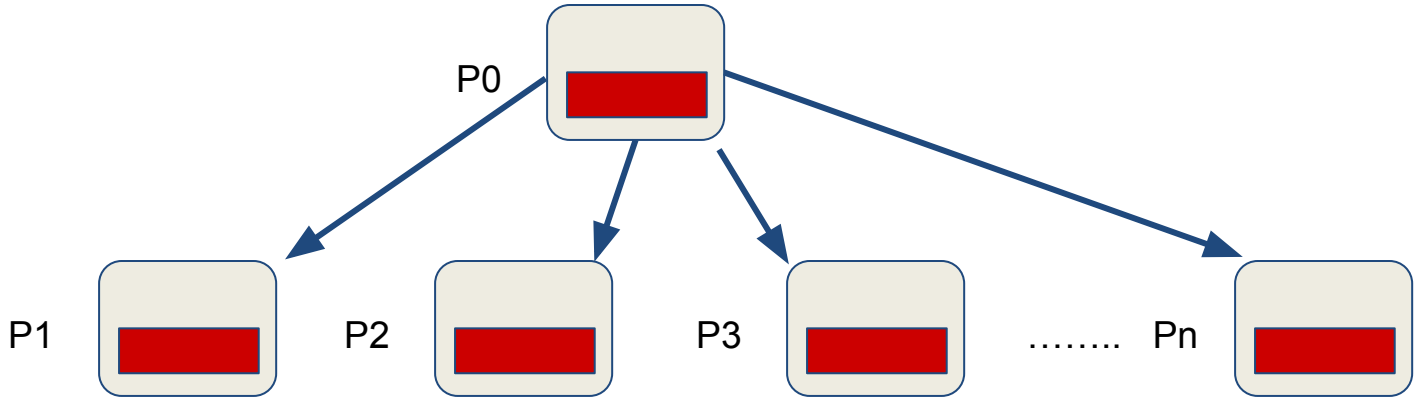


MPI - Broadcast

Syntax :

MPI_Bcast (void* data , Int count , MPI_Datatype datatype , Int source_process , MPI_Comm comm) ;

- One process sends the same data to all processes in a communicator.



MPI - Broadcast : Example

```
void Get input(int my rank ,Int comm_sz , double a_p , double  b_p , int*  n_p )
        {
                if (my rank == 0)
                {
                        printf("Enter a, b, and n \n");
                        scanf("%lf %lf %d", a_p, b_p, n_p);
                }

            MPI Bcast(a_p, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

            MPI Bcast(b_p, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

            MPI Bcast(n_p, 1, MPI _INT, 0, MPI_COMM_WORLD);
        }
```

MPI - Reduce

Syntax :

MPI_Reduce (void* input_data , void* output_data , Int count ,
MPI_Datatype datatype , MPI_Op operator , Int
Dest_process, MPI_Comm comm) ;

MPI - Reduce

Syntax :

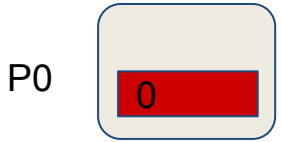
MPI_Reduce (void* input_data , void* output_data , Int count , MPI_Datatype datatype , MPI_Op operator , Int Dest_process, MPI_Comm comm) ;

MPI_MAX
MPI_MIN
MPI_SUM
MPI_PROD
MPI_LAND
:
:
:

MPI - Reduce

Syntax :

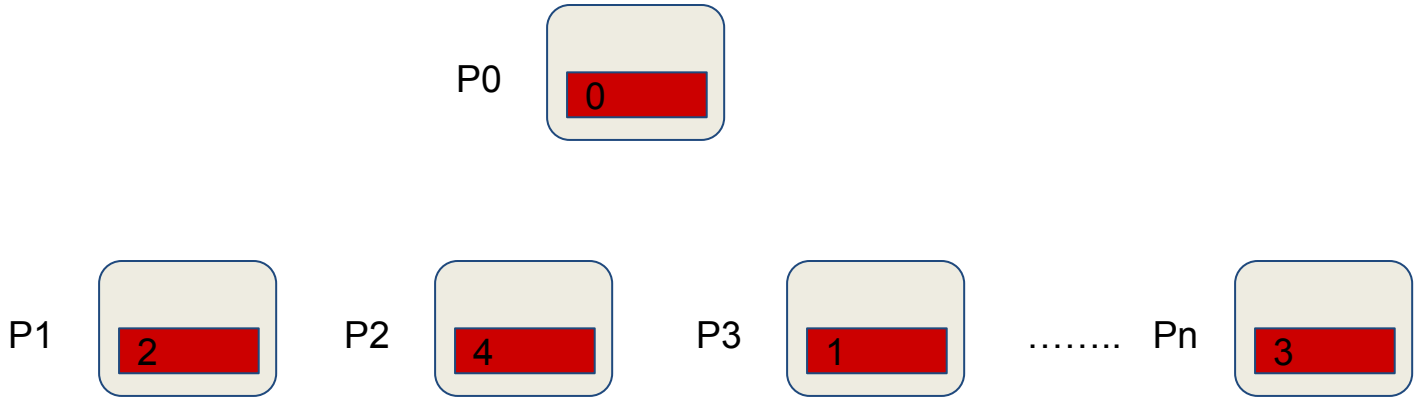
MPI_Reduce (void* input_data , void* output_data , Int count , MPI_Datatype datatype , MPI_Op operator , Int Dest_process, MPI_Comm comm) ;



MPI - Reduce

Syntax :

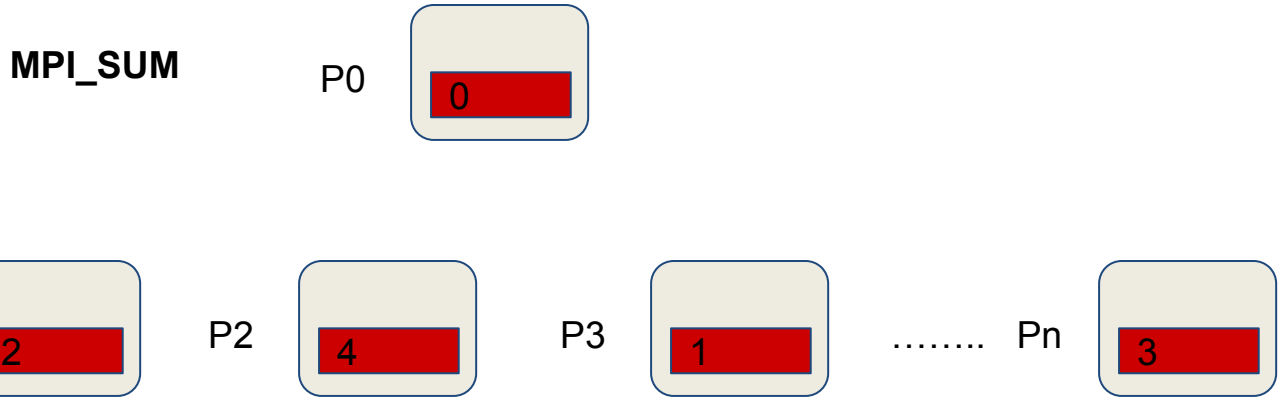
MPI_Reduce (void* input_data , void* output_data , Int count , MPI_Datatype datatype , MPI_Op operator , Int Dest_process, MPI_Comm comm) ;



MPI - Reduce

Syntax :

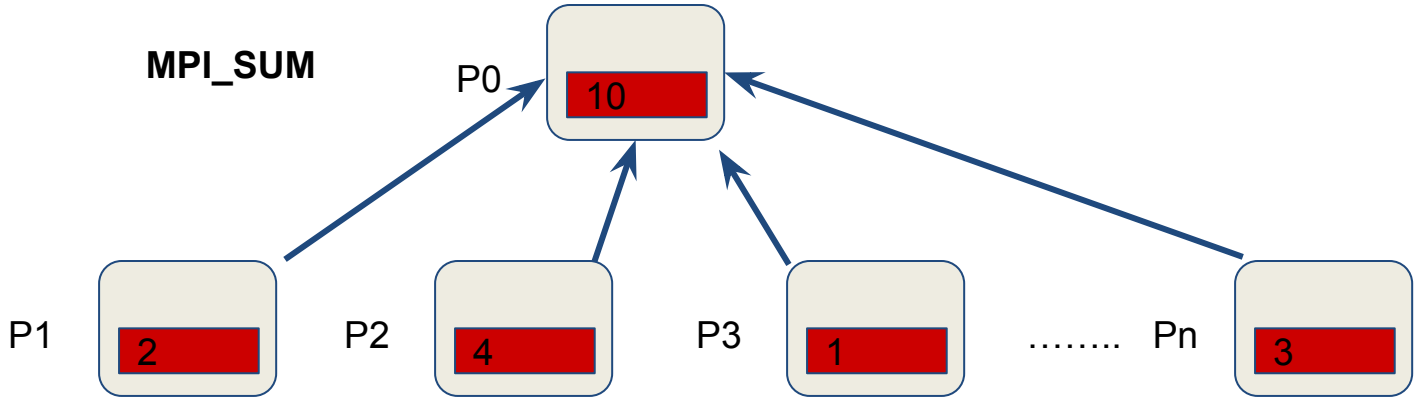
MPI_Reduce (void* input_data , void* output_data , Int count , MPI_Datatype datatype , MPI_Op operator , Int Dest_process, MPI_Comm comm) ;



MPI - Reduce

Syntax :

MPI_Reduce (void* input_data , void* output_data , Int count , MPI_Datatype datatype , MPI_Op operator , Int Dest_process, MPI_Comm comm) ;



MPI - Reduce

Syntax :

MPI_Reduce (void* input_data , void* output_data , Int count , MPI_Datatype datatype , MPI_Op operator , Int Dest_process, MPI_Comm comm) ;

Example : Many lines in Trap. example programs are replaced by this single line ...

MPI - Reduce

Syntax :

MPI_Reduce (void* input_data , void* output_data , Int count , MPI_Datatype datatype , MPI_Op operator , Int Dest_process, MPI_Comm comm) ;

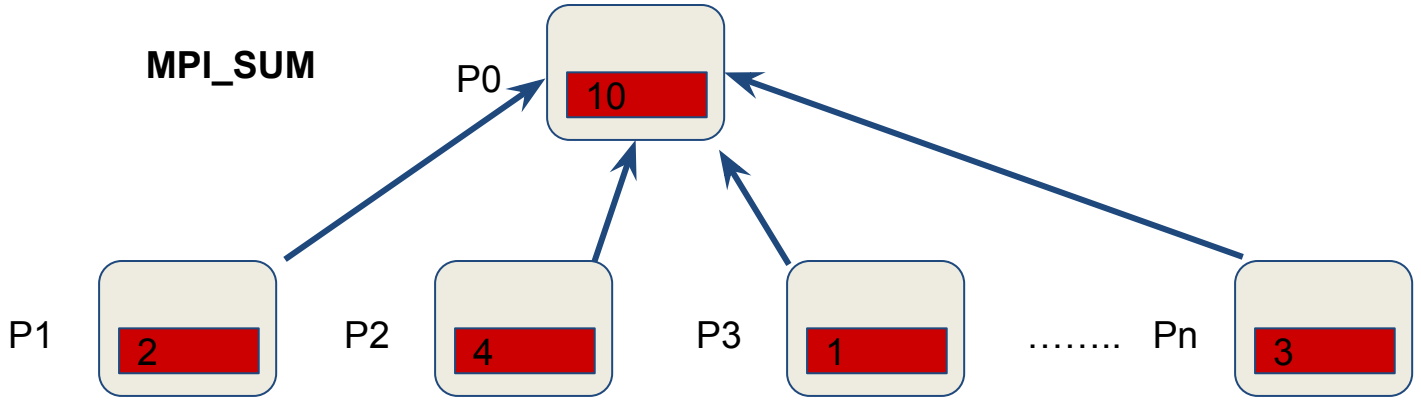
Example : Many lines in Trap. example programs are replaced by this single line ...

MPI_Reduce(&local_int, &total_int, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD) ;

MPI - Allreduce

Syntax :

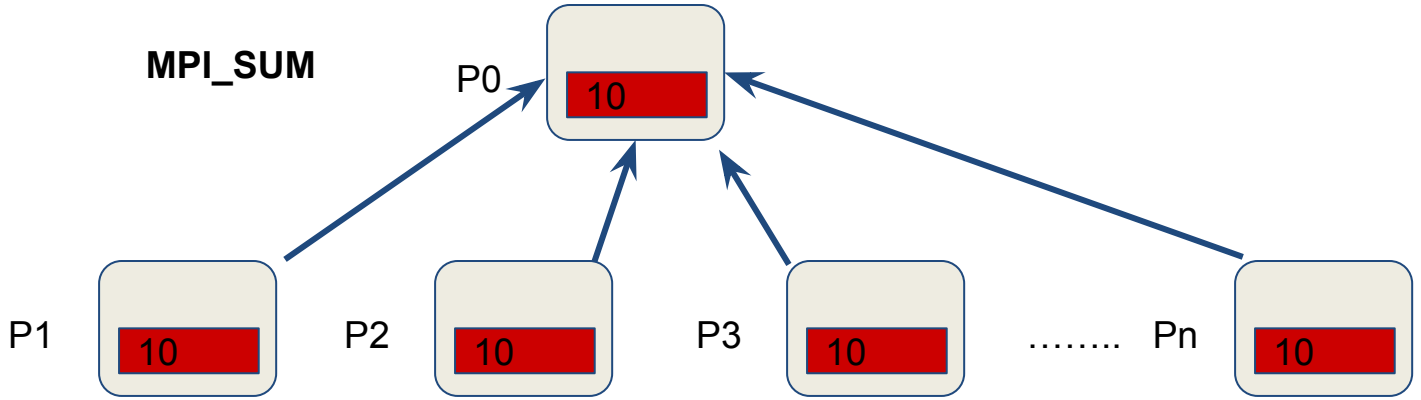
MPI_Reduce (void* input_data , void* output_data , Int count , MPI_Datatype datatype , MPI_Op operator , MPI_Comm comm) ;



MPI - Allreduce

Syntax :

MPI_Reduce (void* input_data , void* output_data , Int count , MPI_Datatype datatype , MPI_Op operator , MPI_Comm comm) ;

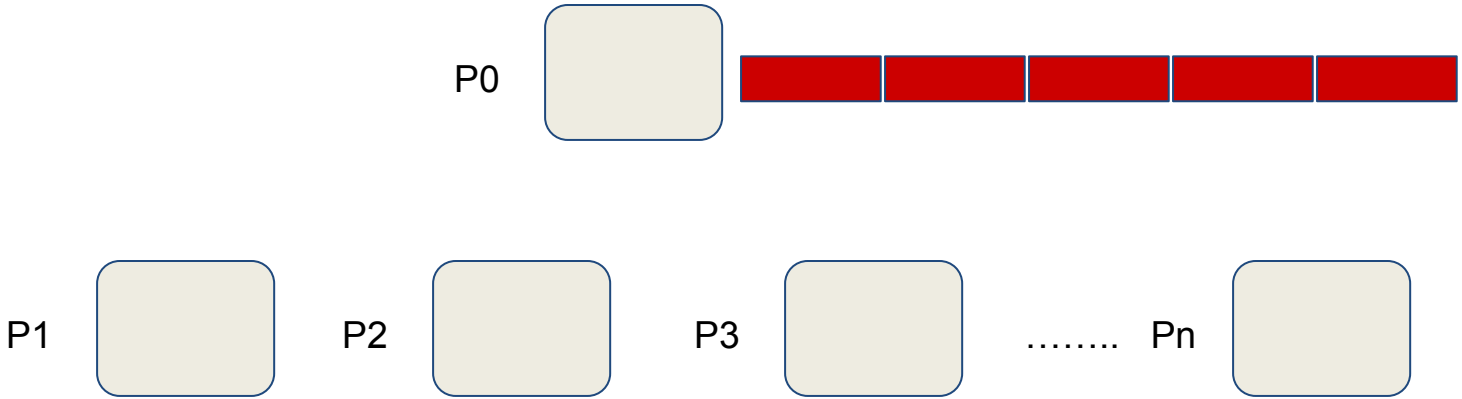


MPI - Scatter

Syntax :

```
MPI_Scatter (void* send_buffer , Int send_count , MPI_Datatype
             send_datatype , void* recv_buffer , Int recv_count ,
             MPI_Datatype recv_datatype , Int source_process ,
             MPI_Comm comm ) ;
```

- MPI_Scatter sends chunks of data to different processes..

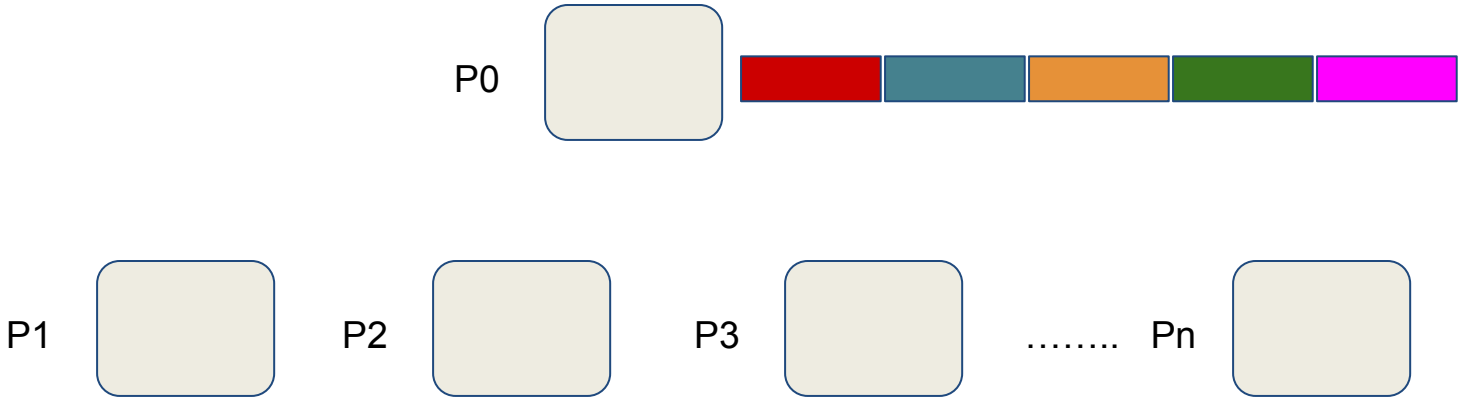


MPI - Scatter

Syntax :

```
MPI_Scatter (void* send_buffer , Int send_count , MPI_Datatype
             send_datatype , void* recv_buffer , Int recv_count ,
             MPI_Datatype recv_datatype , Int source_process ,
             MPI_Comm comm ) ;
```

- MPI_Scatter sends chunks of data to different processes..

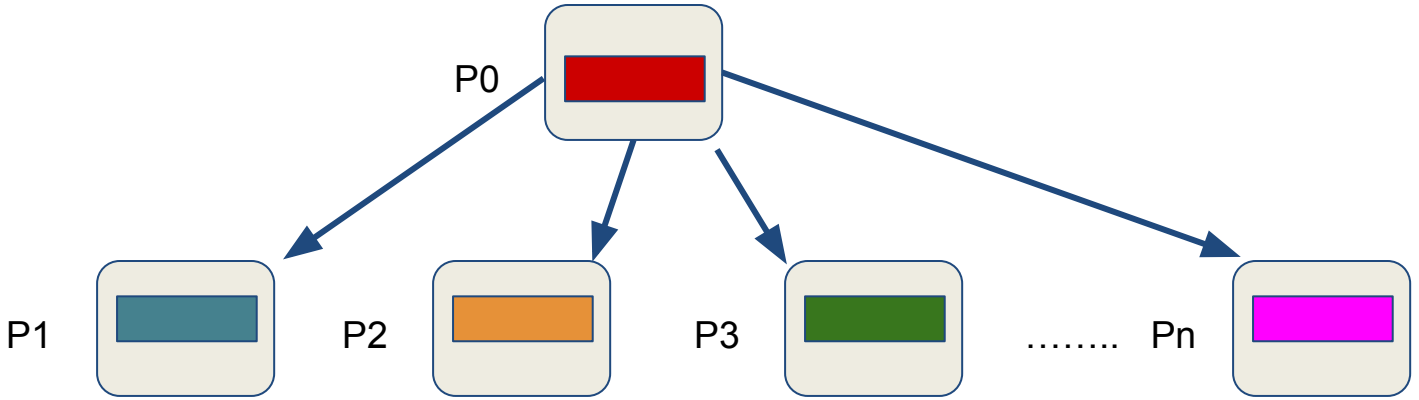


MPI - Scatter

Syntax :

```
MPI_Scatter (void* send_buffer , Int send_count , MPI_Datatype
             send_datatype , void* recv_buffer , Int recv_count ,
             MPI_Datatype recv_datatype , Int source_process ,
             MPI_Comm comm ) ;
```

- MPI_Scatter sends chunks of data to different processes..

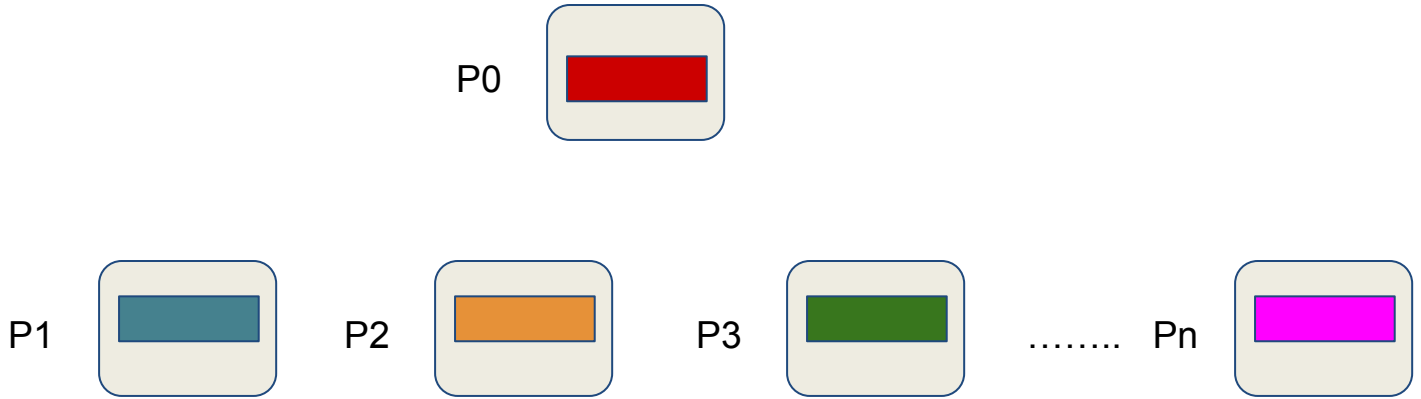


MPI - Gather

Syntax :

```
MPI_Gather (void* send_buffer , Int send_count , MPI_Datatype
            send_datatype , void* recv_buffer , Int recv_count ,
            MPI_Datatype recv_datatype , Int destination_process ,
            MPI_Comm comm ) ;
```

- MPI_Gather collects chunks of data from different processes..

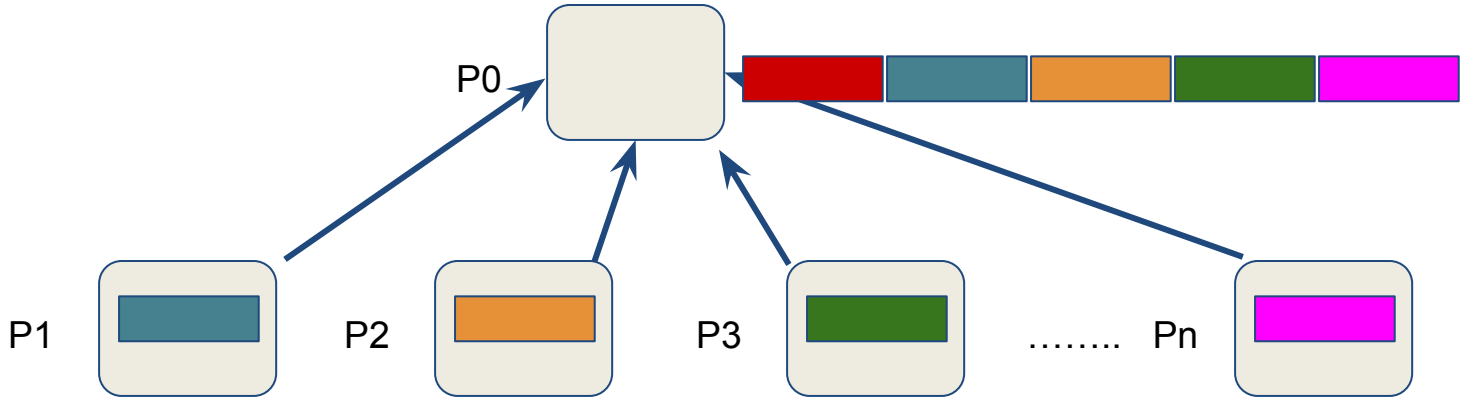


MPI - Gather

Syntax :

```
MPI_Gather (void* send_buffer , Int send_count , MPI_Datatype
            send_datatype , void* recv_buffer , Int recv_count ,
            MPI_Datatype recv_datatype , Int destination_process ,
            MPI_Comm comm ) ;
```

- MPI_Gather collects chunks of data from different processes..

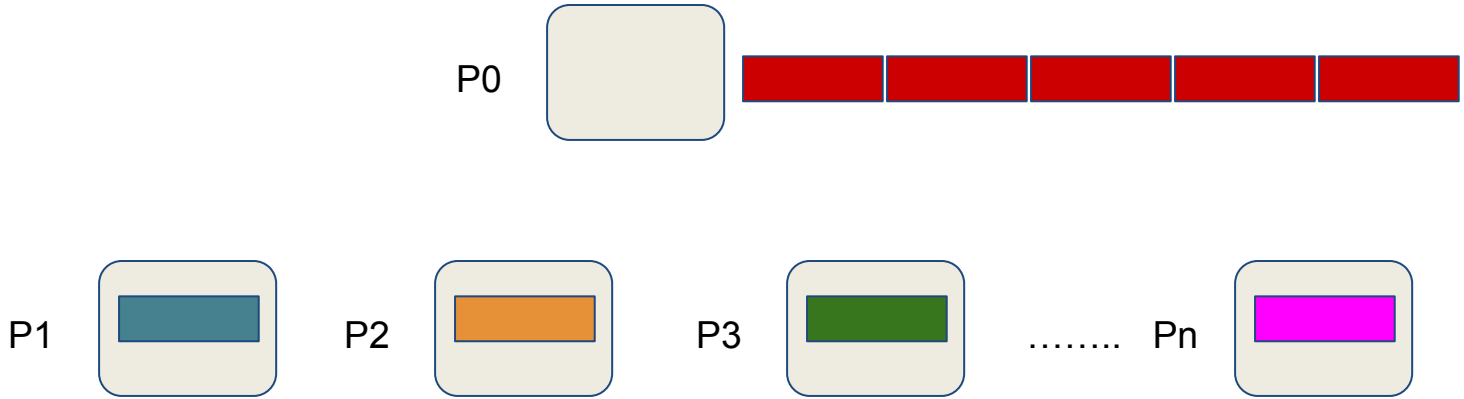


MPI - Gather

Syntax :

```
MPI_Gather (void* send_buffer , Int send_count , MPI_Datatype
            send_datatype , void* recv_buffer , Int recv_count ,
            MPI_Datatype recv_datatype , Int destination_process ,
            MPI_Comm comm ) ;
```

- MPI_Gather collects chunks of data from different processes..



MPI - Allgather

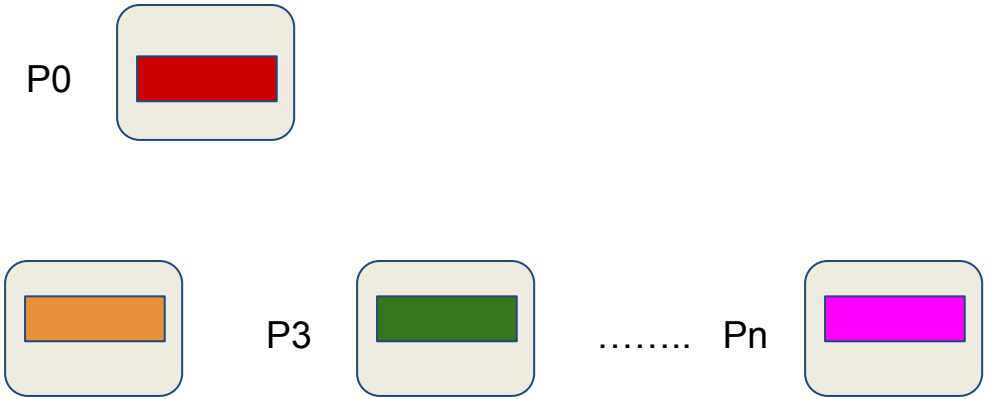
Syntax :

```
MPI_Gather (void* send_buffer , Int  send_count ,  MPI_Datatype
            send_datatype , void*  recv_buffer  , Int  recv_count ,
            MPI_Datatype  recv_datatype ,              ,
            MPI_Comm   comm );
```

MPI - Allgather

Syntax :

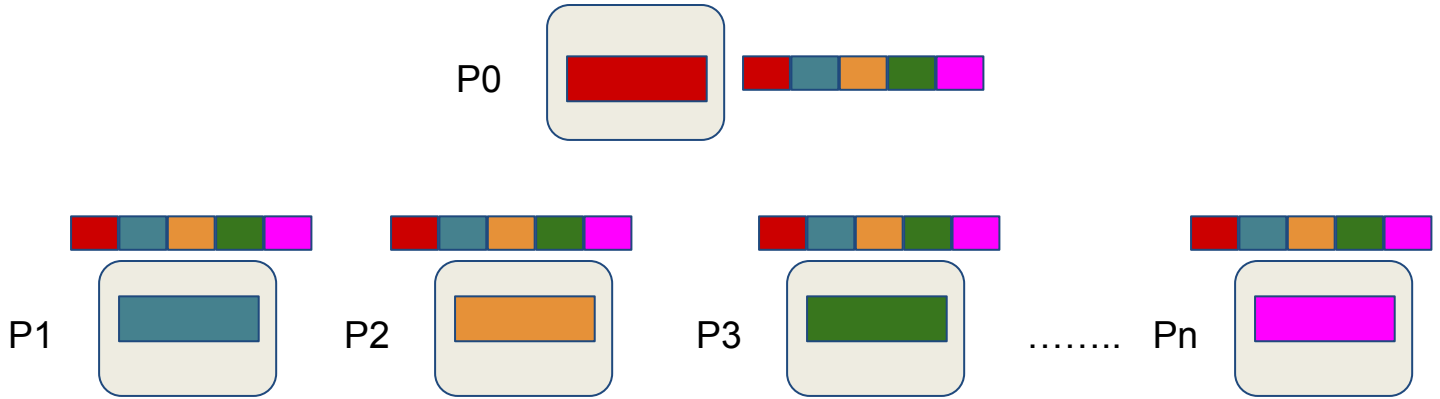
```
MPI_Gather (void* send_buffer , Int send_count , MPI_Datatype
            send_datatype , void* recv_buffer , Int recv_count ,
            MPI_Datatype recv_datatype ,          ,
            MPI_Comm comm );
```



MPI - Allgather

Syntax :

```
MPI_Gather (void* send_buffer , Int send_count , MPI_Datatype
            send_datatype , void* recv_buffer , Int recv_count ,
            MPI_Datatype recv_datatype ,          ,
            MPI_Comm comm );
```



MPI - Synchronization

MPI - Barrier

Syntax :

- **MPI_Barrier (MPI_Comm communicator) ;**

MPI - Barrier

Syntax :

MPI_Barrier (MPI_Comm communicator) ;

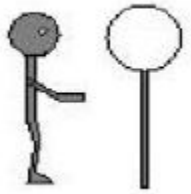
- Used to block the calling process until all processes have entered the function. The call will return at any process only after all the processes or group members have entered the call

- The MPI_BARRIER routine blocks the calling process until all group processes have called the function. When MPI_BARRIER returns, all processes are synchronized at the barrier

MPI - Barrier

Syntax :

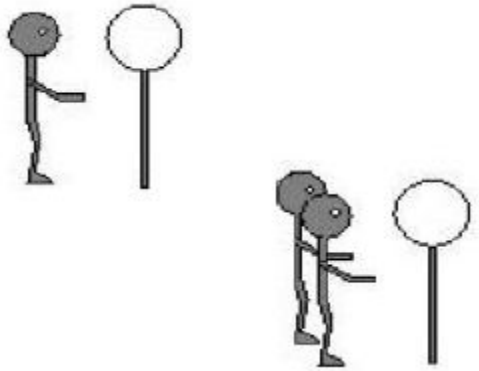
- **MPI_Barrier (MPI_Comm communicator) ;**



MPI - Barrier

Syntax :

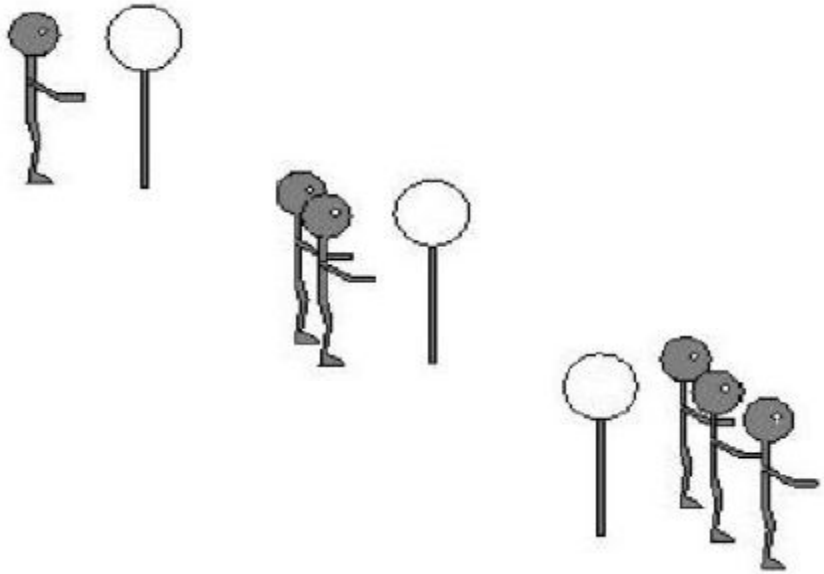
MPI_Barrier (MPI_Comm communicator) ;



MPI - Barrier

Syntax :

MPI_Barrier (MPI_Comm communicator) ;



Recap :

- Point to Point Vs Collective communication -
- MPI_Broadcast(...)
- MPI_Scatter(...)
- MPI_Reduce(...)
- MPI_Allreduce(...)
- MPI_Gather(...)
- MPI_Allgather(...)
- Miss MPI routines !
- ....

References :

[1] Barker, Brandon. "Message passing interface (mpi)." *Workshop: High Performance Computing on Stampede*. Vol. 262. 2015.

[2] Yuan, Chung-Tsz, and Shenjian Chen. "Message Passing Interface (MPI)." (1996).

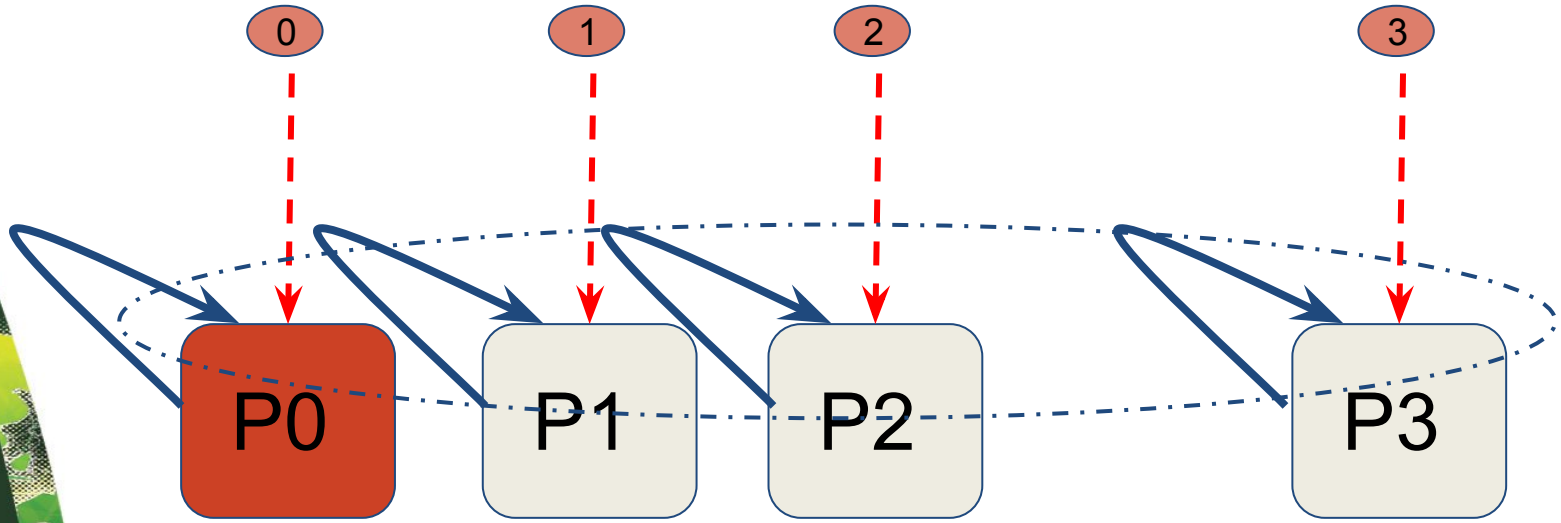
[3] https://computing.llnl.gov/tutorials/mpi/


Thank
You

MPI_Comm_rank(.....)

Syntax :

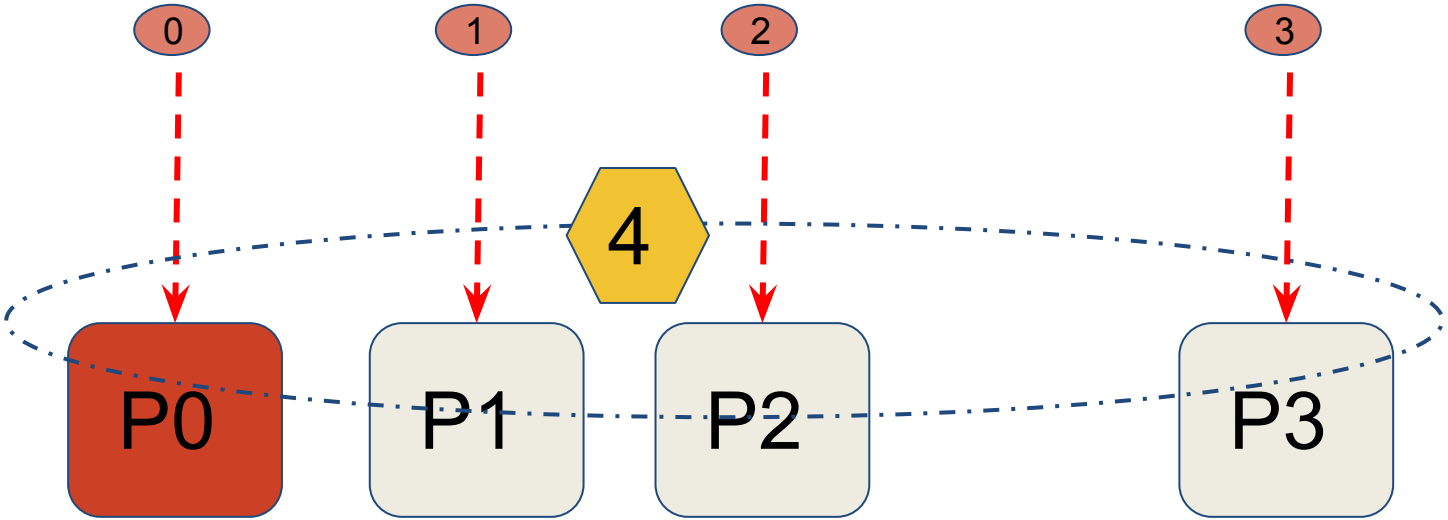
MPI_Comm_rank (MPI_Comm communicator , int * rank) ;



MPI_Comm_size(.....)

Syntax :

MPI_Comm_size (MPI_Comm communicator , int * size) ;



MPI_Send(.....)

Syntax :

1 2 3

```
MPI_Send (void*  msg_buffer , Int  msg_size, MPI_Datatype  msg_type,
          Int destination,  Int  tag , MPI_Comm  communicator ) ;
```

4 5 6

1. Address of Message buffer
2. Message size
3. Data Type
4. Destination process rank
5. Tag - Message Identifier/..
6. Communicator

MPI_Recv(.....)

Syntax :

```
MPI_Recv (void*  msg_buffer , Int  buf_size, MPI_Datatype  buf_type,
          Int source,  Int  tag , MPI_Comm  communicator, MPI_Status* );
```

1 2 3 4 5 6 7

1. Address of Message buffer
2. Buffer size
3. Data Type
4. Source process rank
5. Tag - Message Identifier/..
6. Communicator
7. Status of Received message

❑ Collective Communication Routines

- Used for performing operation on all processes simultaneously
- Approx -- 16

⇨ MPI_Reduce (

```
void*          input_data ,
void*          output_data ,
Int            count  ,
MPI_Datatype   datatype ,
MPI_Op         operator ,
Int            dest_process,
MPI_Comm       comm
) ;
```

MPI_MAX
MPI_MIN
MPI_SUM
MPI_PROD
MPI_LAND
:
:
:

Ex : MPI_Reduce(&local_int, &total_int, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD) ;

⇨ MPI_Allreduce (

```
void*          input_data ,
void*          output_data ,
Int            count  ,
MPI_Datatype   datatype ,
MPI_Op         operator ,

MPI_Comm       comm
) ;
```

Ex : MPI_Allreduce(&local_int, &total_int, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD) ;

Collective Communication Routines

MPI_Bcast (

```
void*           data ,
Int             count ,
MPI_Datatype    datatype ,
Int             source_process ,
MPI_Comm        comm
) ;
```

Eg : MPI_Bcast(a, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD) ;

MPI_Scatter (

```
void*           send_buffer ,
Int             send_count ,
MPI_Datatype    send_datatype ,
void*           recv_buffer  ,
Int             recv_count ,
MPI_Datatype    recv_datatype ,
Int             source_process ,
MPI_Comm        comm
) ;
```

Eg : MPI_Scatter (a, local_n, MPI_DOUBLE, local_a, local_n, MPI_DOUBLE, 0, comm) ;

Collective Communication Routines

- MPI_Gather (

```
        void*           send_buffer ,
        Int             send_count ,
        MPI_Datatype    send_datatype ,
        void*           recv_buffer  ,
        Int             recv_count ,
        MPI_Datatype    recv_datatype ,
        Int             destination_process ,
        MPI_Comm        comm
        ) ;
```

Eg : MPI_Gather (local_b, local_n, MPI_DOUBLE, b, local_n, MPI_DOUBLE, 0, comm) ;

- MPI_Allgather (

```
        void*           send_buffer ,
        Int             send_count ,
        MPI_Datatype    send_datatype ,
        void*           recv_buffer  ,
        Int             recv_count ,
        MPI_Datatype    recv_datatype ,

        MPI_Comm        comm
        ) ;
```

Eg : MPI_Allgather (local_b, local_n, MPI_DOUBLE, b, local_n, MPI_DOUBLE, comm) ;