Collective Communications

Professor Seokkoo Kang
Department of Civil & Environmental Engineering
kangsk78@hanyang.ac.kr

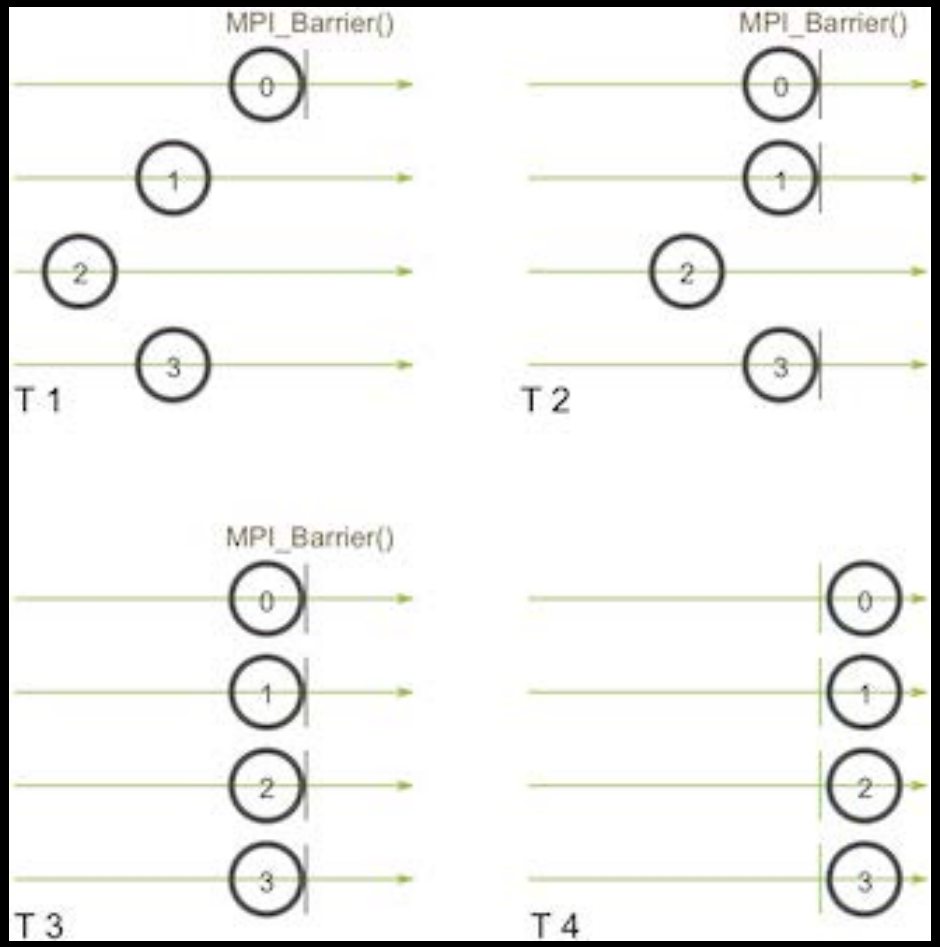
Collective Communication

- All process participate in the communication together. Must be called simultaneously.
- MPI_Barrier
- MPI_Bcast
- MPI_Reduce
- MPI_Allreduce
- MPI_Gather
- MPI_Scatter

MPI_Barrier

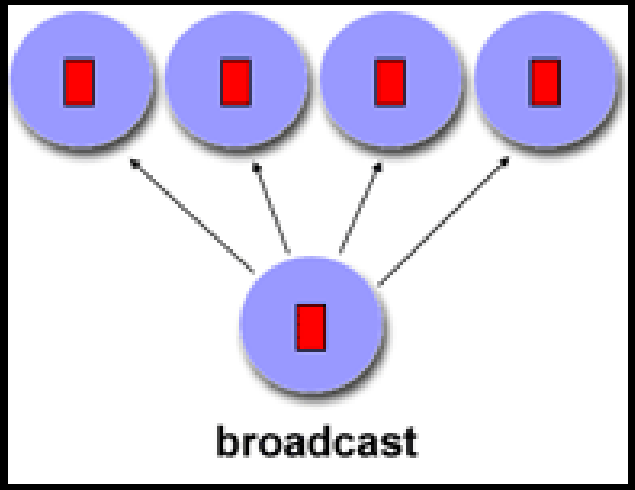
- MPI_Barrier(MPI_Comm communicator)

  Synchronize all processes



MPI_Bcast (Broadcast)

- Distribute the same information from the root process to all others
- MPI_Bcast(void* data, int count, MPI_Datatype datatype, int **root**, MPI_Comm communicator)



MPI_Bcast Example

```
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);

if(rank==0){
        for(i=0;i<10;i++)
        buffer[i]=i*2;
}
MPI_Bcast(buffer, 10, MPI_INT, 0, MPI_COMM_WORLD);
if(rank!=0){
        for(i=0;i<10;i++)
                printf("Process %d: buffer[%d]=%d\n", rank, i, buffer[i]);
}
MPI_Finalize();
```

Output

Process 1: buffer[0]=0
Process 1: buffer[1]=2
Process 1: buffer[2]=4
Process 1: buffer[3]=6
Process 1: buffer[4]=8
Process 1: buffer[5]=10
Process 1: buffer[6]=12
Process 1: buffer[7]=14
Process 1: buffer[8]=16
Process 1: buffer[9]=18

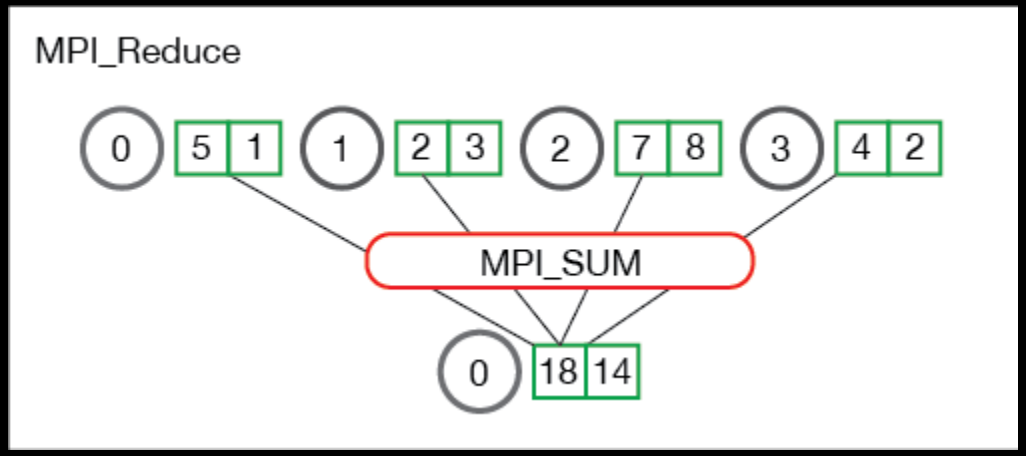
MPI_Reduce

Reduces values on all processes to a single value

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

MPI_Op: MPI_SUM, MPI_MAX, MPI_MIN, MPI_PROD

Root: the core # to which the result is sent.



Sample MPI code

```
MPI_Comm_rank (MPI_COMM_WORLD, &rank);
MPI_Comm_size (MPI_COMM_WORLD, &size);
...
int my_sum=0;
int total_sum=0;
for (i=n_start; i < n_end; i++) my_sum = my_sum + i;
MPI_Reduce(&my_sum, &total_sum, 1, MPI_INT,
                    MPI_SUM,0, MPI_COMM_WORLD);
if(rank==0) printf("#%d, Total sum=%d\n", total_sum);
if(rank==1) printf("#%d, Total sum=%d\n", total_sum);
...
```

Output

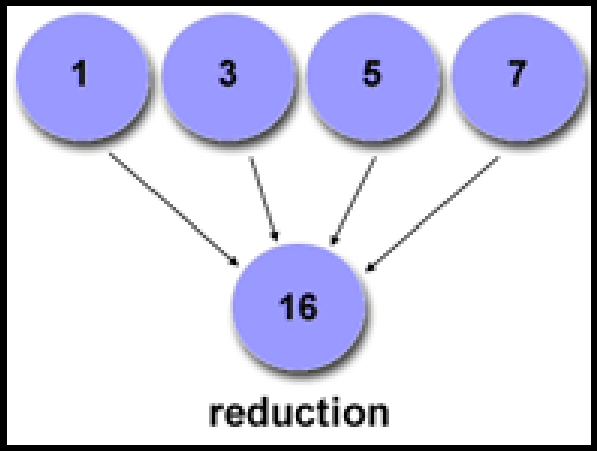
#0, Total sum=5050

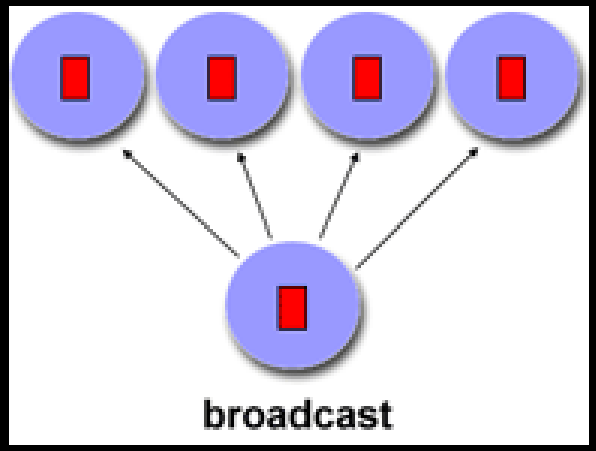
#1, Total sum=0

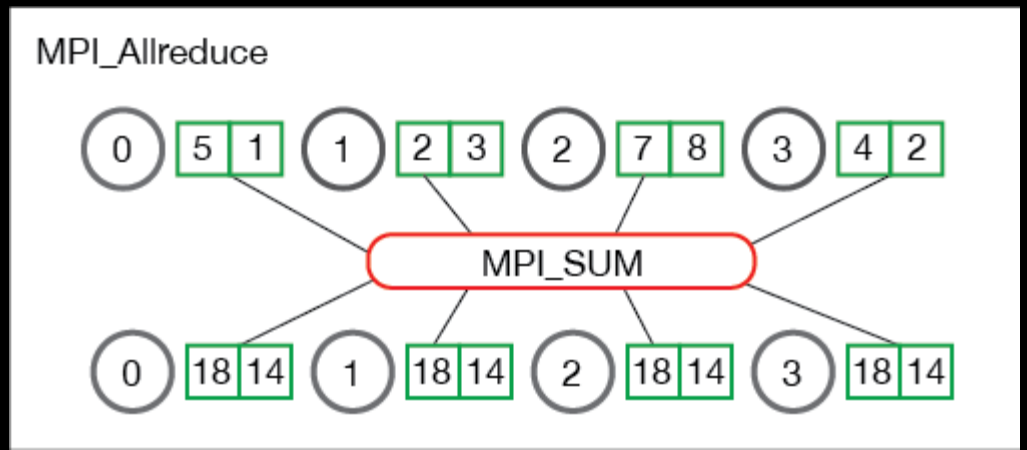
MPI_Allreduce

Reduce + Broadcast

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







Sample MPI code

```
MPI_Comm_rank (MPI_COMM_WORLD, &rank);
MPI_Comm_size (MPI_COMM_WORLD, &size);
...
my_sum=0;
total_sum=0;
for (i=n_start; i < n_end; i++) my_sum = my_sum + i;
MPI_Allreduce(&my_sum, &total_sum, 1, MPI_INT,
                        MPI_SUM, MPI_COMM_WORLD);
if(rank==0) printf("#%d, Total sum=%d\n", total_sum);
if(rank==1) printf("#%d, Total sum=%d\n", total_sum);
...
```

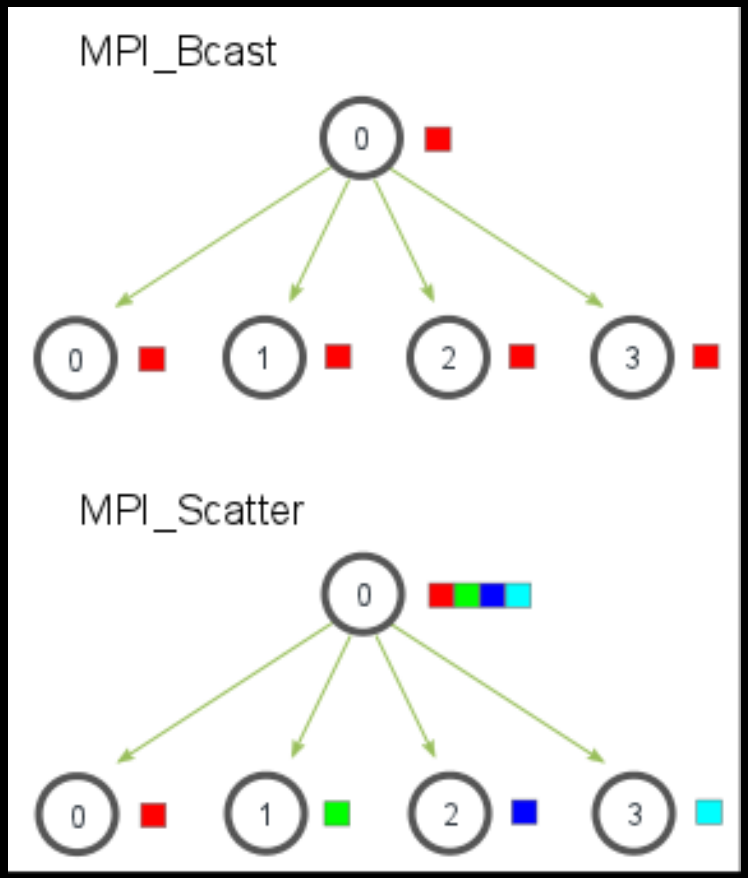
Output

#0, Total sum=5050

#1, Total sum=5050

MPI_Scatter

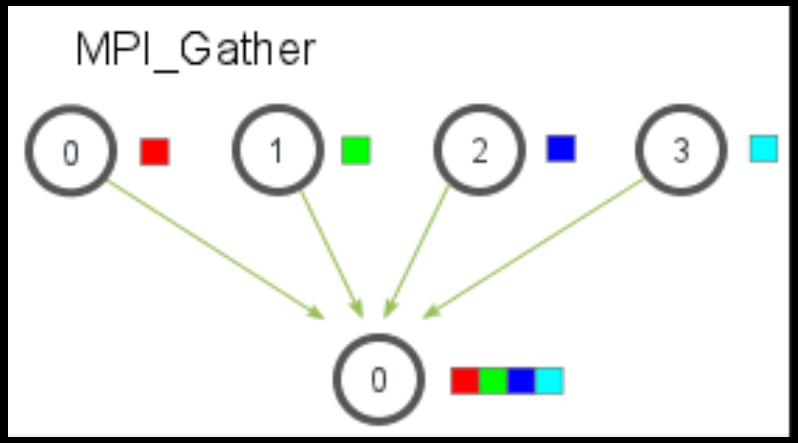
- Splits the data from one process (=root) to all other processes (compare it with MPI_Bcast)



MPI_Scatter (void* send_data, int send_count, MPI_Datatype send_datatype, void* recv_data, int recv_count, MPI_Datatype recv_datatype, int root, MPI_Comm communicator)

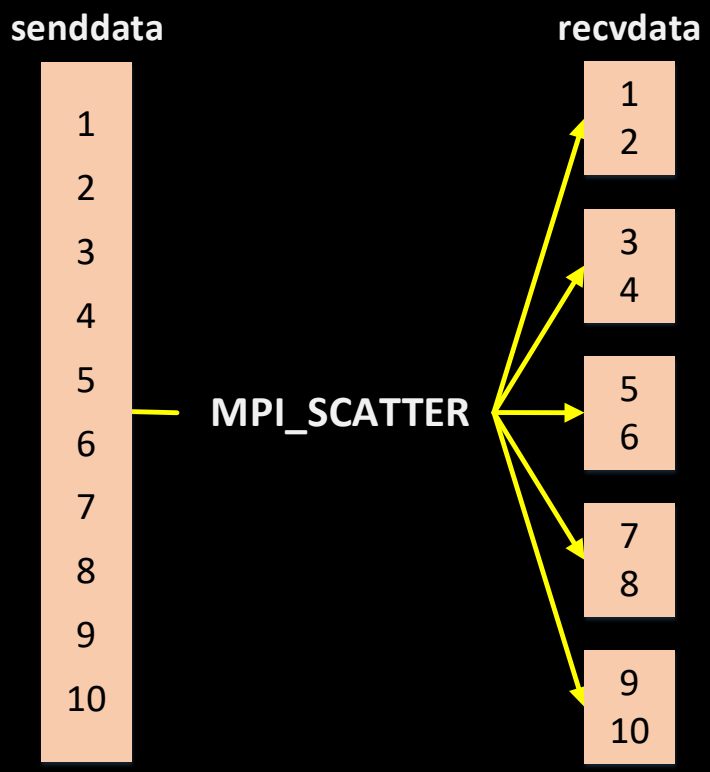
MPI_Gather

- Collects the data from each process and stores the data in process rank order in recv_data on the root.



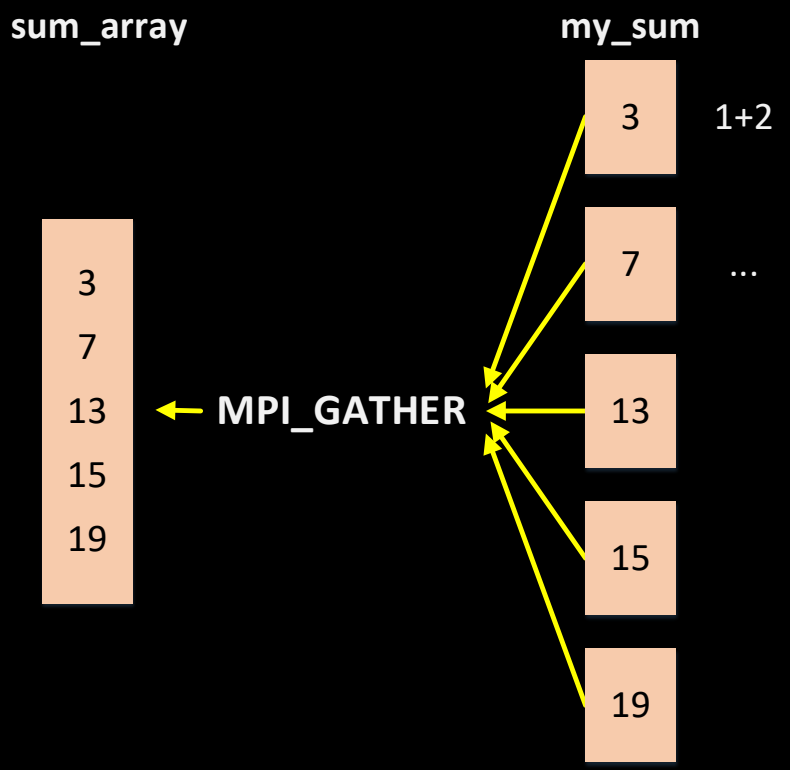
MPI_Gather (void* send_data, int send_count, MPI_Datatype send_datatype, void* recv_data, int recv_count, MPI_Datatype recv_datatype, int root, MPI_Comm communicator)

MPI_Scatter Example




```
totalrank=5; /*assume total process=5 */
int senddata[10], recvdata[2];
int sendsize=2, recvsize=2;
if (rank==0) for(i=0; i<10; i++) senddata[i]=i+1;
MPI_Scatter(senddata, sendsize, MPI_INT,
            recvdata, recvsize, MPI_INT, 0,
            MPI_COMM_WORLD);
```

MPI_Gather Sample Code

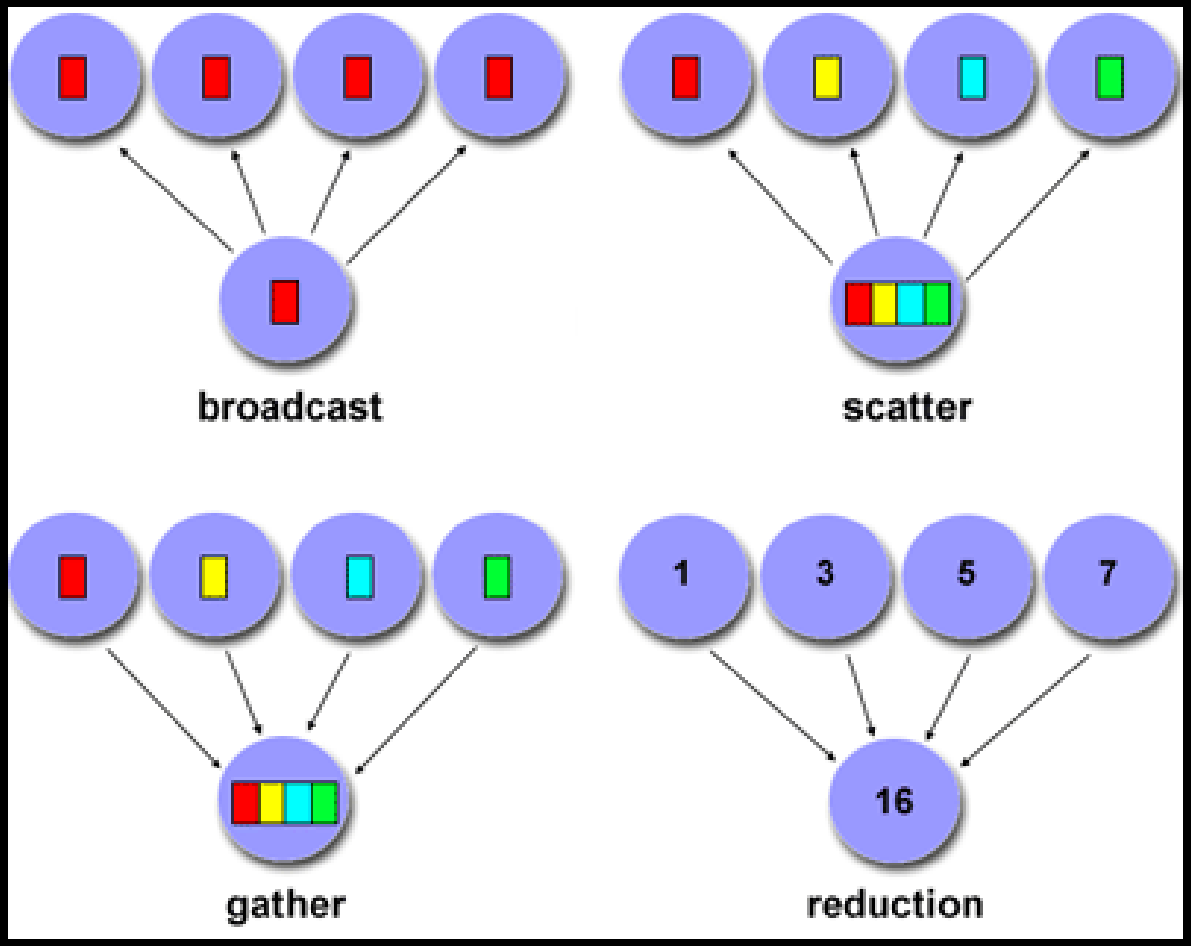



```
int my_sum=0;
for(int i=0; i<recvsize; i++)
        my_sum = my_sum + recvdata[i];
int *sum_array = malloc( sizeof(int) *total_rank );
MPI_Gather (  &my_sum, 1, MPI_INT,
                        &sum_array[0], 1, MPI_INT,
                        0, MPI_COMM_WORLD);
```

```
MPI_Gather (  void* send_data, int send_count, MPI_Datatype        send_datatype,
              void* recv_data, int recv_count, MPI_Datatype  recv_datatype,
              int root, MPI_Comm communicator)
```

Vairants

- MPI_Scatterv: variable length
- MPI_Gatherv
- MPI_Allgather (MPI_Gather+MPI_Bcast)


broadcast
scatter
gather
1
3
5
7
16
reduction

Assignment

- Write a code for calculating the sum from 1 to 10,000 and printing the result in two different ways.
  1. Use only MPI_Scatter and MPI_Gather.
  2. Use only MPI_Reduce.
- Upload the ppt material after each presentation.