Distributed Computing

MPI

Mahmoud Badry, mma18@fayoum.edu.eg

Collective Operations

Collective/Global Communication Operations

Mahmoud Badry, mma18@fayoum.edu.eg

Collective/Global Communication Operations

- A communication operation is called collective or global if all or a subset of the processes of a parallel program are involved.

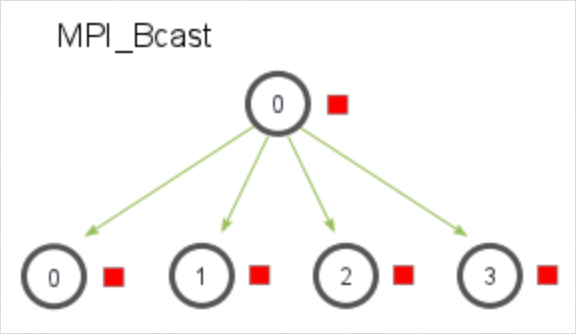
Global communication operation	MPI function
Broadcast operation	`MPI_Bcast()`
Accumulation operation	`MPI_Reduce()`
Gather operation	`MPI_Gather()`
Scatter operation	`MPI_Scatter()`
Multi-broadcast	`MPI_Allgather()`
Multi-accumulation operation	`MPI_Allreduce()`

- Collective MPI communication operations are always blocking; no non-blocking versions are provided as is the case for point-to-point operations

Mahmoud Badry, mma18@fayoum.edu.eg

Broadcast operation

- For a broadcast operation, one specific process of a group of processes sends the same data block to all other processes of the group.
-
```
int MPI_Bcast (void *message,
               int count,
               MPI_Datatype type,
               int root,
               MPI_Comm comm)
```



- Where **root** denotes the process which sends the data block.
- Each process must specify the same root process and must use the same communicator.
- Data blocks sent by `MPI_Bcast()` cannot be received by an `MPI_Recv()`.

Mahmoud Badry, mma18@fayoum.edu.eg

Broadcast operation cont.

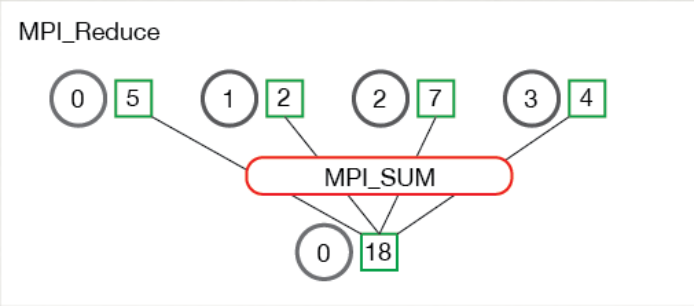
- The MPI runtime system guarantees that broadcast messages are received in the same order in which they have been sent by the root process, even if the corresponding broadcast operations are not executed at the same time.

```
if (my_rank == 0) {
   MPI_Bcast (&x, 1, MPI_INT, 0, comm);
   MPI_Bcast (&y, 1, MPI_INT, 0, comm);
   local_work ();
}
else if (my_rank == 1) {
   local_work ();
   MPI_Bcast (&y, 1, MPI_INT, 0, comm);
   MPI_Bcast (&x, 1, MPI_INT, 0, comm);
}
else if (my_rank == 2) {
   local_work ();
   MPI_Bcast (&x, 1, MPI_INT, 0, comm);
   MPI_Bcast (&y, 1, MPI_INT, 0, comm);
}
```


Mahmoud Badry, mma18@fayoum.edu.eg

Reduction Operation

- For such an operation, each participating process provides a block of data that is combined with the other blocks using a binary reduction operation. The accumulated result is collected at a root process.
-
```
int MPI_Reduce (void *sendbuf,
                void *recvbuf,
                int count,
                MPI_Datatype type,
                MPI_Op op,
                int root,
                MPI_Comm comm)
```



- The parameter **recvbuf** specifies the receive buffer which is provided by the root process.

Mahmoud Badry, mma18@fayoum.edu.eg

Reduction Operation Cont.

Representation	Operation
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Sum
MPI_PROD	Product
MPI_LAND	Logical and
MPI_BAND	Bit-wise and
MPI_LOR	Logical or
MPI_BOR	Bit-wise or
MPI_LXOR	Logical exclusive or
MPI_BXOR	Bit-wise exclusive or
MPI_MAXLOC	Maximum value and corresponding index
MPI_MINLOC	Minimum value and corresponding index

Mahmoud Badry, mma18@fayoum.edu.eg

Reduction Operation Cont.

- For an **`MPI_Reduce()`** operation, all participating processes must specify the same values for the parameters **`count`**, **`type`**, **`op`**, and **`root`**. The send buffers **`sendbuf`** and the receive buffer **`recvbuf`** must have the same size.
- MPI supports the definition of user-defined reduction operations using the following MPI function:
  -
```
int MPI_Op_create (MPI_User function *function,
                   int commute,
                   MPI_Op *op)
```


Mahmoud Badry, mma18@fayoum.edu.eg

Reduction Operation Cont.

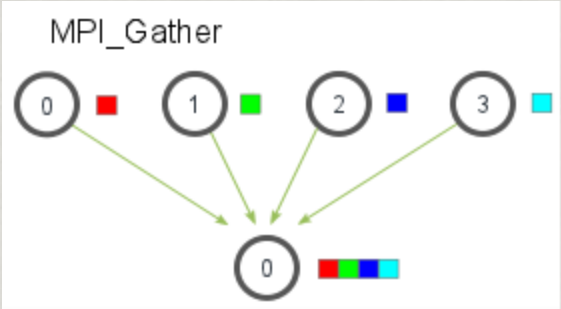
```
double ain[30], aout[30];
int ind[30];
struct {double val; int rank;} in[30], out[30];
int i, my_rank, root=0;

MPI_Comm_rank (MPI_COMM_WORLD, &my_rank);
for (i=0; i<30; i++) {
   in[i].val = ain[i];
   in[i].rank = my_rank;
}
MPI_Reduce(in,out,30,MPI_DOUBLE_INT,MPI_MAXLOC,root,MPI_COMM_WORLD);
if (my_rank == root)
   for (i=0; i<30; i++) {
      aout[i] = out[i].val;
      ind[i] = out[i].rank;
   }
```


Mahmoud Badry, mma18@fayoum.edu.eg

Gather Operation

- For a gather operation, each process provides a block of data collected at a root process. Like reduce but with no operation which make the receive buffer bigger.
- ```
  int MPI_Gather(void *sendbuf,
                 int sendcount,
                 MPI_Datatype sendtype,
                 void *recvbuf,
                 int recvcount,
                 MPI_Datatype recvtype,
                 int root,
                 MPI_Comm comm).
  ```

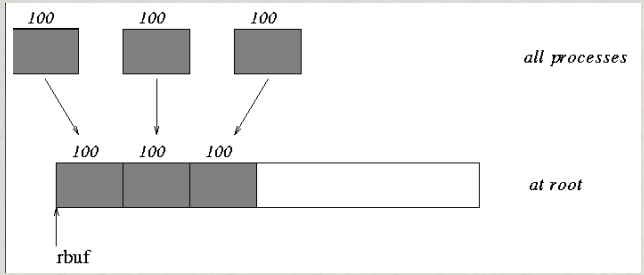


- They are stored in the order of the ranks of the processes according to comm.

Mahmoud Badry, mma18@fayoum.edu.eg
  ```

Gather Operation cont.

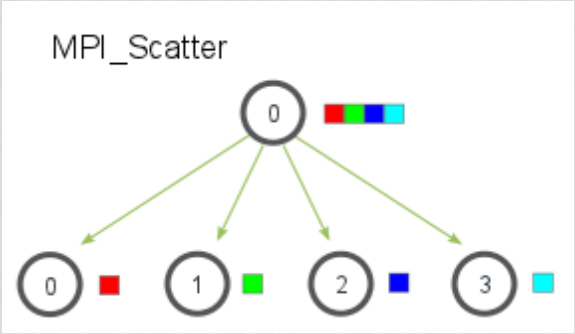
```
MPI_Comm comm;
int sendbuf[100], my_rank, root = 0, gsize, *rbuf;
MPI_Comm_rank (comm, &my_rank);
if (my_rank == root) {
   MPI_Comm_size (comm, &gsize);
   rbuf = (int *) malloc (gsize*100*sizeof(int));
}
MPI_Gather(sendbuf, 100, MPI_INT, rbuf, 100, MPI_INT, root, comm);
```



Mahmoud Badry, mma18@fayoum.edu.eg

Scatter Operation

- For a scatter operation, a root process provides a different data block for each participating process.
-
```
int MPI_Scatter (void *sendbuf,
                 int sendcount,
                 MPI_Datatype sendtype,
                 void *recvbuf,
                 int recvcount,
                 MPI_Datatype recvtype,
                 int root,
                 MPI_Comm comm)
```

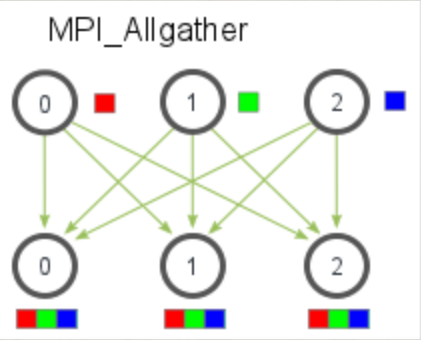


- They are scattered in the order of the ranks of the processes according to comm.

Mahmoud Badry, mma18@fayoum.edu.eg

Multi-broadcast Operation (MPI_Allgather)

- Each participating process contributes a block of data which could, for example, be a partial result from a local computation
-
```
int MPI_Allgather (void *sendbuf,
                   int sendcount,
                   MPI_Datatype sendtype,
                   void *recvbuf,
                   int recvcount,
                   MPI_Datatype recvtype,
                   MPI_Comm comm)
```



- They are gathered in the order of the ranks of the processes according to comm.

Mahmoud Badry, mma18@fayoum.edu.eg

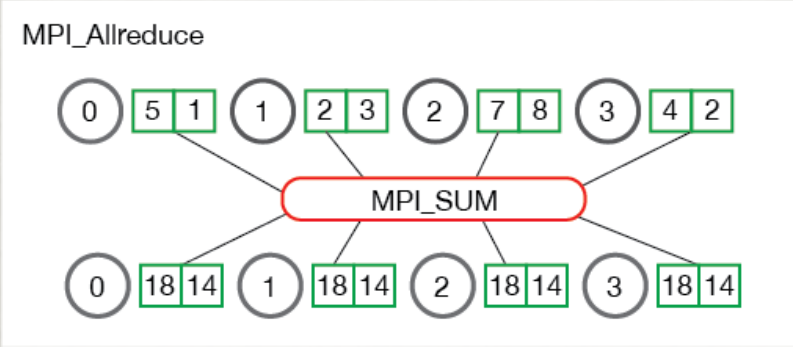
Multi-broadcast Operation (MPI_Allgather) cont.

```
int sbuf[100], gsize, *rbuf;
MPI_Comm_size (comm, &gsize);
rbuf = (int*) malloc (gsize*100*sizeof(int));
MPI_Allgather (sbuf, 100, MPI_INT, rbuf, 100, MPI_INT, comm);
```


Mahmoud Badry, mma18@fayoum.edu.eg

Multi-accumulation Operation (MPI_Allreduce)

- Each participating process performs a separate single-accumulation operation for which each process provides a different block of data.
-
```
int MPI_Allreduce (void *sendbuf,
                   void *recvbuf,
                   int count,
                   MPI_Datatype type,
                   MPI_Op op,
                   MPI_Comm comm)
```



Mahmoud Badry, mma18@fayoum.edu.eg

QUESTIONS

Mahmoud Badry, mma18@fayoum.edu.eg

THANK YOU!

Mahmoud Badry, mma18@fayoum.edu.eg