Collective Operations with MPI

Henry Neeman, Josh Alexander, Andrew Fitz Gibbon and Charlie Peck

OU Supercomputing Symposium Workshop
SC Education Program
October, 2009

```
#include "fancy logos and distracting graphics"
```

How Did We Get Here?

Point–to–point communications are the fundemental building–blocks of distributed memory parallel programs, however there are other more powerful constructs available which can make the design and implementation of parallel programs easier.

Those constructs are generally known as collective operations since they involve communication between groups of processes rather than a pair of individual ones.

Overview of the Collective Operations

- All–To–One
  - `MPI_Gather()`, `MPI_Gatherv()`, `MPI_Reduce()`
- One–To–All
  - `MPI_Bcast()`, `MPI_Scatter()`, `MPI_Scatterv()`
- All–To–All
  - `MPI_Allgather()`, `MPI_Allgatherv()`, `MPI_Allreduce()`
- Other
  - `MPI_Barrier()`

All–To–One

- `MPI_Gather()` - Collect the same amount of data from each process in a communicator (including the root).

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

- `MPI_Gatherv()` - Collect a varying amount of data from each process in a communicator (including the root).

```
int MPI_Gatherv(void *sendbuf, int sendcount, MPI_Datatype
sendtype, void *recvbuf, int *recvcounts, int *displs,
MPI_Datatype recvtype, int root, MPI_Comm comm)
```

- `MPI_Reduce()` - Combines the elements provided in the input buffer of each process in the group, using the operation op, and returns the combined value in the output buffer of the process with rank root.

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

One–To–All

- `MPI_Bcast()` - Broadcasts a message from the process with rank root to all processes of the group, itself included. It is called by all members of group using the same arguments for comm and root.

  ```
  int MPI_Bcast(void *buffer, int count, MPI_Datatype
  datatype, int root, MPI_Comm comm)
  ```

- `MPI_Scatter()` - MPI_Scatter is the inverse operation to MPI_Gather, it sends data from one task to all tasks in a group.

  ```
  int MPI_Scatter(void *sendbuf, int sendcount, MPI_Datatype
  sendtype, void *recvbuf, int recvcount, MPI_Datatype
  recvtype, int root, MPI_Comm comm)
  ```

- `MPI_Scatterv()` - MPI_Scatterv is the inverse operation to MPI_Gatherv, it extends the functionality of MPI_Scatter by allowing a varying count of data to be sent to each process, since sendcounts is now an array.

```
int MPI_Scatterv(void *sendbuf, int *sendcounts, int
*displs, MPI_Datatype sendtype, void *recvbuf, int
recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
```

All–To–All

- `MPI_Allgather()` - MPI_Allgather is similar to MPI_Gather, except that all processes receive the result, instead of just the root. In other words, all processes contribute to the result, and all processes receive the result.

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

- `MPI_Allgatherv()` - MPI_Allgatherv is similar to MPI_Allgather in that all processes gather data from all other processes, except that each process can send a different amount of data. The block of data sent from the jth process is received by every process and placed in the jth block of the buffer recvbuf.

  ```
  int MPI_Allgatherv(void *sendbuf, int sendcount,
  MPI_Datatype sendtype, void *recvbuf, int *recvcount, int
  *displs, MPI_Datatype recvtype, MPI_Comm comm)
  ```

- `MPI_Allreduce()` - Combines the elements provided in the input buffer of each process in the group, using the operation op, and returns the combined value in the output buffer of all processes.

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

Other

- `MPI_Barrier()` - Blocks the caller until all group members have called it; the call returns at any process only after all group members have entered the call.

  ```
  int MPI_Barrier(MPI_Comm comm)
  ```

Questions?

Lab – Using MPI's Collective Operations