Skip to main content

- Home
- Topics
- Reference
- Glossary
- Help
- Notebook

Virtual Workshop

Welcome guest
Log in (Globus)
Log in (other)
Try the quiz before you start
MPI Collective Communications
Introduction Goals Prerequisites
Characteristics Three Types of Routines Barrier Synchronization Data Movement • Broadcast • Gather and Scatter • Gather/Scatter Effect • Gatherv and Scatterv • Allgather • All to All Global Computing • Reduce • Scan • Operations and Example • Allreduce Mini-Exercise Nonblocking Routines • Nonblocking Example Performance Issues • Two Ways to Broadcast • Two Ways to Scatter Application Example • Scatter vs. Scatterv • Scatterv Syntax
Exercise Quiz
Short survey

MPI Collective Communications: Reduce

One of the most useful collective operations is a global reduction, which is a combined-result type of operation. The outcome from applying some desired function across all processes in the group is collected in one specified process or in all the processes. If there are n processes in the process group, and d(i,j) is the j-th data item in process i, then the D(j) item of data in the root process that will be returned from a reduce routine is given by:

$$D(j) = d(0,j)*d(1,j)* \ ... \ *d(n-1,j)$$

where * is the reduction function, which is always assumed associative. All MPI predefined functions are also assumed to be commutative. Each process can provide either one element or a sequence of elements. In both cases the reduce operation is executed element-wise on each element of the sequence (index j above).

There are three versions of reduce. They are MPI_Reduce, MPI_Allreduce, and MPI_Reduce_scatter. The form of these reduction primitives is listed below:

C

```
int MPI_Reduce(const void* sbuf, void* rbuf, int count, \
     MPI_Datatype stype, MPI_Op op, int root, MPI_Comm comm)

int MPI_Allreduce(const void* sbuf, void* rbuf, int count \
     MPI_Datatype stype, MPI_Op op, MPI_Comm comm)

int MPI_Reduce_scatter_block(const void* sbuf, void* rbuf, \
     int rcount, MPI_Datatype stype, MPI_Op op, MPI_Comm comm)

int MPI_Reduce_scatter(const void* sbuf, void* rbuf, \
```

```
const int[] rcount, MPI_Datatype stype, MPI_Op op, \
MPI_Comm comm)
```

FORTRAN

```
MPI_REDUCE(sbuf, rbuf, count, stype, op, root, comm, ierr)

MPI_ALLREDUCE(sbuf, rbuf, count, stype, op, comm, ierr)

MPI_REDUCE_SCATTER(sbuf, rbuf, rcount, stype, op, comm, ierr)
```

The differences among these three reduces:

- MPI_Reduce returns results to a single process;
- MPI_Allreduce returns results to all processes in the group;
- MPI_Reduce_scatter_block scatters a vector of results from a reduce operation across all processes, in blocks of the same size.
- MPI_Reduce_scatter scatters a vector of results from a reduce operation across all processes, in blocks of variable size.

In the above:

sbuf	is the address of send buffer,
rbuf	is the address of receive buffer,
count	is the number of elements in send buffer, OR
rcount	are the numbers of elements to be scattered back,
stype	is the data type of elements of send buffer,
op	is the reduce operation (MPI predefined, or your own),
root	is the rank of the root process,
comm	is the group communicator.

Notes:

- `rbuf` is significant only at the root process for MPI_Reduce.
- The `rcount` argument in MPI_Reduce_scatter is an array; even though `count` doesn't appear explicitly in this call, it is equal to the sum of the elements in `rcount`.
- While an MPI_REDUCE followed by an MPI_SCATTER (or MPI_SCATTERV) is functionally similar to MPI_REDUCE_SCATTER_BLOCK (or MPI_REDUCE_SCATTER), the MPI implementations likely have optimized these combined functions to be more efficient, and so they should be used where possible. Also note the exception to the similarity: MPI_REDUCE_SCATTER can't have displacements in its send buffer, which is probably why these functions do not use the previous nomenclature where the pair of functions would be called MPI_REDUCE_SCATTER and MPI_REDUCE_SCATTER***V***.

<= previous next =>

Add my notes
Mark (M) my place in this topic
