MPI_ALLREDUCE, MPI_Allreduce

Purpose

Applies a reduction operation to the vector **sendbuf** over the set of tasks specified by *comm* and places the result in *recvbuf* on all of the tasks in *comm*.

C synopsis

```
#include <mpi.h>
int MPI_Allreduce(void* sendbuf,void* recvbuf,int count,
  MPI_Datatype datatype,MPI_Op op,MPI_Comm comm);
```

C++ synopsis

```
#include mpi.h
void MPI::Comm::Allreduce(const void* sendbuf, void* recvbuf, int count,
  const MPI::Datatype& datatype, const MPI::Op& op)
  const;
```

FORTRAN synopsis

```
include 'mpif.h' or use mpi
MPI_ALLREDUCE(CHOICE SENDBUF,CHOICE RECVBUF,INTEGER COUNT,
      INTEGER DATATYPE,INTEGER OP,INTEGER COMM,INTEGER IERROR)
```

Parameters

sendbuf
is the starting address of the send buffer (choice) (IN)

recvbuf
is the starting address of the receive buffer (choice) (OUT)

count
is the number of elements in the send buffer (integer) (IN)

datatype
is the datatype of elements in the send buffer (handle) (IN)

op is the reduction operation (handle) (IN)

comm
is the communicator (handle) (IN)

IERROR
is the FORTRAN return code. It is always the last argument.

Description

This subroutine applies a reduction operation to the vector **sendbuf** over the set of tasks specified by *comm* and places the result in *recvbuf* on all of the tasks.

This subroutine is similar to MPI_REDUCE except the result is returned to the receive buffer of all the group members.

The ″in place″ option for intracommunicators is specified by passing the value MPI_IN_PLACE to the argument *sendbuf* at the root. In this case, the input data is taken at each task from the receive buffer, where it will be replaced by the output data.

If *comm* is an intercommunicator, the result of the reduction of the data provided by tasks in group A is stored at each task in group B, and vice versa. Both groups should provide the same count value.

MPI_IN_PLACE is not supported for intercommunicators.

When you use this subroutine in a threads application, make sure all collective operations on a particular communicator occur in the same order at each task. See *IBM Parallel Environment for AIX: MPI Programming Guide* for more information on programming with MPI in a threads environment.

Notes

See *IBM Parallel Environment for AIX: MPI Programming Guide* for information about reduction functions.

The MPI standard urges MPI implementations to use the same evaluation order for reductions every time, even if this negatively affects performance. PE MPI adjusts its reduce algorithms for the optimal performance on a given task distribution. The MPI standard suggests, but does not mandate, this sacrifice of performance. PE MPI chooses to put performance ahead of the MPI standard's recommendation. This means that two runs with the same task count may produce results that differ in the least significant bits, due to rounding effects when evaluation order changes. Two runs that use the same task count and the same distribution across nodes will always give identical results.

Errors

Fatal errors:

Invalid count
count < **0**

Invalid datatype

Type not committed

Invalid op

Invalid communicator

Unequal message lengths

Invalid use of MPI_IN_PLACE

MPI not initialized

MPI already finalized

Develop mode error if:

Inconsistent op

Inconsistent datatype

Inconsistent message length

Related information

MPE_IALLREDUCE
MPI_OP_CREATE
MPI_REDUCE
MPI_REDUCE_SCATTER