MPI

-
-

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Young Won Lim
06/18/2012

The Butterfly Swap Operations

Communicators and Groups defines
collection of processes that may communicate with each other.

Need to specify a communicator as an argument.

MPI_COMM_WORLD - predefined communicator
that includes all of your MPI processes.

Within a communicator, every process has its own unique, integer identifier,
called rank or “task ID”.

Used to specify the source and destination.
Also can be used in conditional statements.

MPI_Alltoall

MPI_Alltoall - Sends data from all to all processes

```
int MPI_Alltoall( void *sendbuf, int sendcount, MPI_Datatype sendtype,
                  void *recvbuf, int recvcnt, MPI_Datatype recvtype, MPI_Comm comm )
```

INPUT PARAMETERS

sendbuf - starting address of send buffer (choice)

sendcounts - integer array equal to the group size specifying the number of elements to send to each processor

sendtype - data type of send buffer elements (handle)

recvcounts - integer array equal to the group size specifying the maximum number of elements that can be received from each processor

recvtype - data type of receive buffer elements (handle)

comm - communicator (handle)

OUTPUT PARAMETERS

recvbuf - address of receive buffer (choice)

MPI_Alltoallv

MPI_Alltoallv - Sends data from all to all processes, with a displacement

int MPI_Alltoallv (void *sendbuf, int *sendcnts, int *sdispls, MPI_Datatype sendtype,
void *recvbuf, int *recvcnts, int *rdispls, MPI_Datatype recvtype, MPI_Comm comm)

INPUT PARAMETERS

sendbuf - starting address of send buffer (choice)

sendcounts - integer array equal to the group size specifying the number of elements to send to each processor

sdispls - integer array (of length group size). Entry j specifies the displacement (relative to sendbuf from which to take the outgoing data destined for process j

sendtype - data type of send buffer elements (handle)

recvcounts - integer array equal to the group size specifying the maximum number of elements that can be received from each processor

rdispls - integer array (of length group size). Entry i specifies the displacement (relative to recvbuf at which to place the incoming data from process

recvtype - data type of receive buffer elements (handle)

comm - communicator (handle)

OUTPUT PARAMETERS

recvbuf - address of receive buffer (choice)

MPI_Alltoallv

Alltoallv
flexibility in that the location of send data is specified by sdispls
and the location of the placement of receive data is specified by rdispls.

The **jth block** sent from **process i** is received
by **process j** and is placed in the **ith block**.

Need not be all the same size block

sendcount[j], sendtype at **process i**
recvcount[i], recvtype at **process j**.

The amount of data sent must be equal to the amount of data received,
pairwise between every pair of processes.

Distinct type maps between sender and receiver are still allowed.

MPI_Alltoallw

ALLTOALLW in MPI-2.

Can specify separately count, displacement, and datatype.

The displacement of blocks is specified in bytes.

Can be seen as a generalization several MPI functions depending on the input arguments.

Eager Protocol

For short messages

The message itself + supplementary info may be sent
and stored at the receiver side
In some buffer space without receiver's intervention

message envelope – length, sender, tag, etc

A matching receiver operation may not be needed
But message must be copied from the intermediate buffer to the receive buffer

+Synchronization overhead is reduced
- May require large amount of preallocated buffer space
- Flooding a process with many eager messages may overflow → contention

Rendezvous Protocol

For large messages

Buffering the data makes no sense

The envelope is immediately stored at the receiver
The actual message transfer blocks until the user's receive buffer is available

Extra data copy could be avoided,
improving effective bandwidth,
but sender and receiver must synchronize.

MPI_Issend

Starts a **nonblocking synchronous** send

int MPI_Issend(void *buf, int count, MPI_Datatype datatype, int dest, int tag,
MPI_Comm comm, MPI_Request *request)

Input Parameters

buf - initial address of send buffer (choice)
count - number of elements in send buffer (integer)
datatype - datatype of each send buffer element (handle)
dest - rank of destination (integer)
tag - message tag (integer)
comm - communicator (handle)

Output Parameter

request - communication request (handle)

References

[1] http://en.wikipedia.org/
[2] http://static.msi.umn.edu/tutorial/scicomp/general/MPI/mpi_coll_new.html
[3] https://computing.llnl.gov/tutorials/mpi/
[4] https://computing.llnl.gov/tutorials/mpi/
[5] Hager & Wellein, Introduction to High Performance Computing for Scientists and Engineers