A Preview of MPI 3.0: The Shape of Things to Come

Manjunath Gorentla Venkata
manjugv@ornl.gov

Joshua Hursey
hurseyjj@ornl.gov

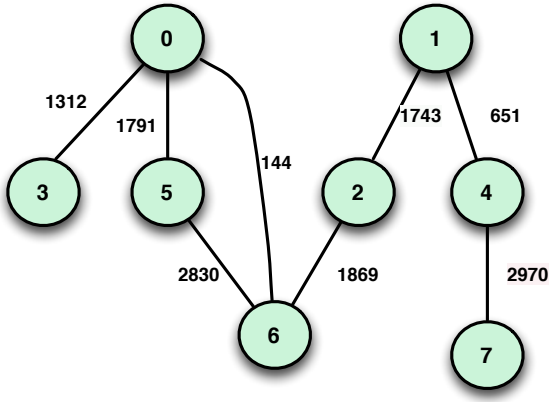
Overview of New Features in MPI 2.2

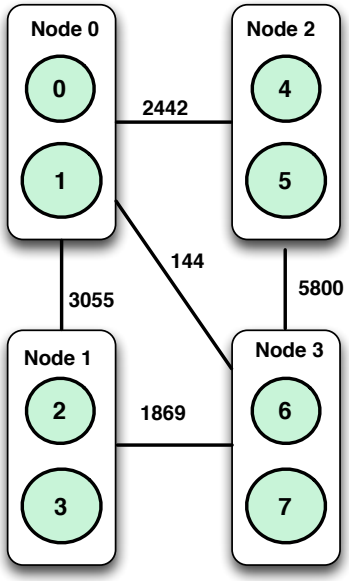
- New functions
- New datatypes
- Minor function updates
- Text changes (not covered)
- Errata (not covered)

Virtual Topologies

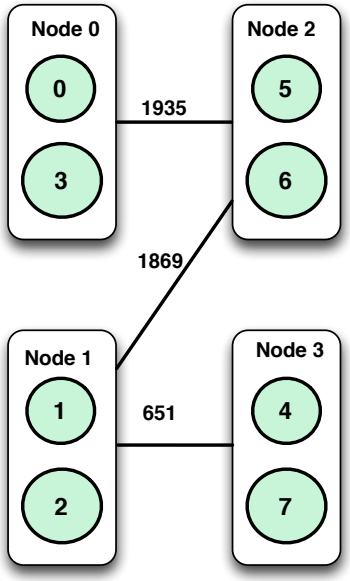
- Virtual Topologies
  - Provides users a convenient way to rename processes
  - Provides users a naming scheme reflecting the communication
  - Provides implementers information to optimize performance for user supplied communication graph

Topology Mapping Example





Naive Mapping



Optimized Mapping

Gottschling and Hoefler: Productive Parallel Linear Algebra Programming with Unstructured Topology Adaption

OAK RIDGE National Laboratory

MPI 2.1 Topology Support

- Cartesian topology
  - Each process is identified by a Cartesian co-ordinate
- Graph topology
  - Process organization is defined by Graphs
  - Each process is represented by a node, and communication between processes is represented by edges

MPI 2.1 Topology Routines

MPI_Cart_create(MPI_Comm comm_old, int ndims, const int *dims, const int *periods, int reorder, MPI_Comm *comm_cart)

- ndims – number of dimensions of Cartesian grid
- dims – number of processes in each dimension
- periods – array specifying whether the grid is periodic (true) or not (false) in each dimension
- reorder – ranking may be reordered (true) or not (false) (logical)

MPI 2.1 Topology Routines

MPI_Graph_create(MPI_Comm comm_old, int nnodes, const int *index, const int *edges, int reorder, MPI_Comm *comm_graph)

- nnodes – Total number of nodes
- index – array storing the neighbors of all previous nodes
- edges – stores edges of the all processes
- reorder – ranking may be reordered (true) or not (false) (logical)

MPI 2.1 Graph Topology Interface is Not Scalable

- Each process stores the whole graph
- Each process requires O(n^2) memory
- Each process requires Ω(n) memory for sparse graphs
- Processes cannot attach communication weights for edges

OAK RIDGE National Laboratory

Distributed Graph Functions

MPI_DIST_GRAPH_CREATE_ADJACENT
(comm_old, indegree, sources, sourceweights, outdegree, destinations, destweights, info, reorder, comm_dist_graph)

- Properties
  - Each process contributes all its incoming and outgoing edges
  - Requires no communication for building the graph
  - Doubles the size of graph
- Example

  Process 0

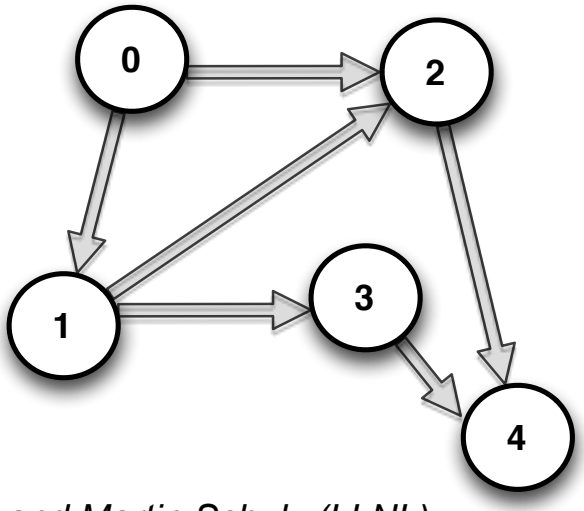
  indegree – 0 sources {}

  outdegree- 2 Destinations {1,2}

  Process 1

  indegree – 1 sources {0}

  outdegree- 2 Destinations {2, 3}



Thanks to Torsten Hoefler (UIUC) and Martin Schulz (LLNL)

OAK RIDGE National Laboratory

Distributed Graph Functions (cont.)

MPI_DIST_GRAPH_CREATE*(comm_old, n, sources, degrees, destinations, weights, info, reorder, comm_dist_graph)*

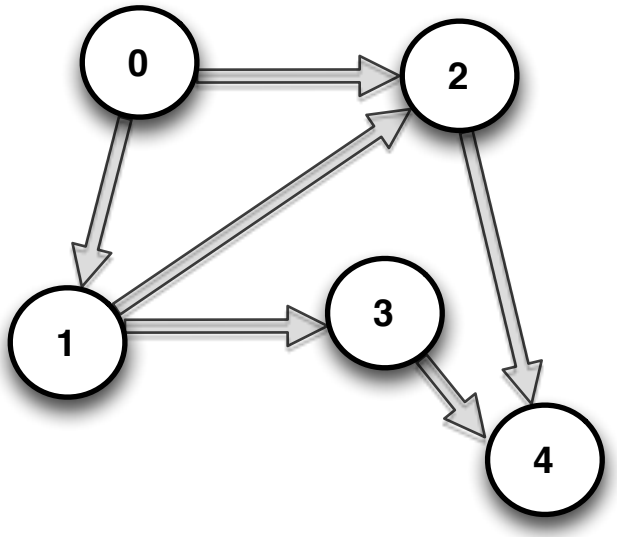
- Properties
  - Each process contributes a subset of the graph
  - Requires communication for building the graph
- Example

Process 0

n : 2
Sources {0,2}
Degrees {1,1}
Destinations {1,4}

Process 1

n : 2
Sources {1,3}
Degrees {1,1}
Destinations {3,4}



Thanks to Torsten Hoefler (UIUC) and Martin Schulz (LLNL)

Adjacency Information Query Functions

MPI_Dist_graph_neighbors_count(MPI_Comm comm, int *indegree, int *outdegree, int *weighted)

- Returns number of neighbors

MPI_Dist_graph_neighbors(MPI_Comm comm, int maxindegree, int sources[], int sourceweights[], int maxoutdegree, int destinations[],int destweights[])

- Returns neighbors list

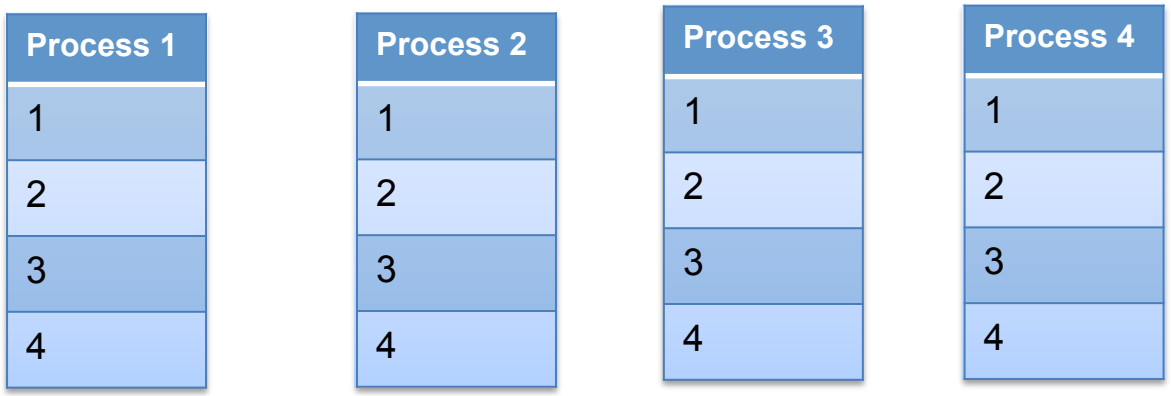
OAK RIDGE National Laboratory

Reduction Operations

MPI_Reduce_scatter_block (void *sendbuf, void *recvbuf, int recvcount, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)

Example of a 4-process reduce scatter block

Data at each process



Data after reduction and scatter operation

OAK RIDGE National Laboratory

Reduction Operations (cont.)

MPI_Reduce_local(void *inbuf, void *inoutbuf, int count, MPI_Datatype datatype, MPI_Op op)

- inbuf – address of input buffer
- inoutbuf – address of input-output buffer
- count - number of elements in each buffer

MPI_Op_commutative(MPI_Op op, int *commute)

- commute - true if op is commutative, false otherwise (logical)

OAK RIDGE National Laboratory

Read Access Restrictions on Send Buffer

- Reading sendbuffer while operation is in progress is valid

- Example

```
MPI_Isend(buff, count, MPI_INT, dest, TAG_ARBITRARY, comm, &request);
fprintf(stdout,"Buffer Value %d", *buff );
MPI_Wait(&request, MPI_STATUS_IGNORE);
```

MPI_INPLACE Valid for More Functions

- MPI_ALLTOALL
- MPI_ALLTOALLV
- MPI_ALLTOALLW
- MPI_EXSCAN

OAK RIDGE National Laboratory

New Predefined Types

- Datatypes to take advantage of changes in C language standard (c99)
  - MPI_(U)INT{8,16,32,64 }_T
  - MPI_C_BOOL
  - MPI_C_FLOAT_COMPLEX
  - MPI_C_DOUBLE_COMPLEX
  - MPI_C_LONG_DOUBLE_COMPLEX

- Datatypes that can be passed between languages without conversion
  - MPI_AINT
  - MPI_OFFSET

Status of C++ Language Bindings

- C++ bindings are deprecated in MPI 2.2
- C++ bindings are going away in MPI 3.0

Summary

- Optimized topology construction
- New reduction operations
- Support for MPI_INPLACE
- Support for new predefined datatypes
- C++ bindings deprecated

Acknowledgments

- Center for Computational Sciences
- MPI Forum
- Torsten Hoefler (UIUC) and Martin Schulz (LLNL)