Lecture 28: Process Topology and MPI

William Gropp

www.cs.illinois.edu/~wgropp
Virtual and Physical Topologies

- A virtual topology represents the way that MPI processes communicate
  - Nearest neighbor exchange in a mesh
  - Recursive doubling in an all-to-all exchange
- A physical topology represents that connections between the cores, chips, and nodes in the hardware
Virtual and Physical Topologies

- Issue is mapping of the virtual topology onto the physical topology
  - Hierarchical systems (e.g., nodes of chips of cores) makes this more complicated; no simple topology
- Questions to ask
  - Does it really matter what mapping is used?
  - How does one get a good mapping?
  - How bad can a bad mapping be?
  - What if the mapping is random?
- This lecture is about using MPI to work with virtual topologies and make it possible for the MPI implementation to provide a good mapping
MPI’s Topology Routines

• MPI provides routines to create new communicators that order the process ranks in a way that may be a better match for the physical topology
• Two types of virtual topology supported:
  ♦ Cartesian (regular mesh)
  ♦ Graph (several ways to define in MPI)
• Additional routines provide access to the defined virtual topology
• (Virtual) topologies are properties of a communicator
  ♦ Topology routines all create a new communicator with properties of the specified virtual topology
MPI Cartesian Topology

- Create a new virtual topology using
  - MPI_Cart_create
- Determine “good” sizes of mesh with
  - MPI_Dims_create
MPI_Cart_create

- MPI_Cart_create(MPI_Comm oldcomm, int ndim, int dims[], int qperiodic[], int qreorder, MPI_Comm *newcomm)

  Creates a new communicator newcomm from oldcomm, that represents an ndim dimensional mesh with sizes dims. The mesh is periodic in coordinate direction i if qperiodic[i] is true. The ranks in the new communicator are reordered (to better match the physical topology) if qreorder is true.
MPI_Dims_create

- MPI_Dims_create(int nnodes, int ndim, int dims[])

- Fill in the dims array such that the product of dims[i] for i=0 to ndim-1 equals nnodes.

- Any value of dims[i] that is 0 on input will be replaced; values that are > 0 will not be changed.
MPI_Cart_create Example

- int periods[3] = {1,1,1};
  int dims[3] = {0,0,0}, wsize;
  MPI_Comm cartcomm;

  MPI_Comm_size(MPI_COMM_WORLD, &wsize);
  MPI_Dims_create(wsize, 3, dims);
  MPI_Cart_create(MPI_COMM_WORLD, 3, dims, periods, 1, &cartcomm);

- Creates a new communicator cartcomm that may be efficiently mapped to the physical topology
Information About a Cartesian Topology

- **MPI_Cartdim_get**
  - Dimension of Cartesian mesh (\texttt{ndim})

- **MPI_Cart_get**
  - Size of dimensions (\texttt{dims}), periodic dimensions (\texttt{qperiodic}), coordinates of calling process in mesh
Determine Neighbor Ranks

• Can be computed from rank (in the cartcomm), dims, and periods, since ordering defined in MPI
  ♦ See Section 7.5 in MPI-3 Standard

• Easier to use either
  ♦ MPI_Cart_coords, MPI_Cart_rank
  ♦ MPI_Cart_shift
MPI_Cart_shift

- MPI_Cart_shift(MPI_Comm comm, int direction, int disp, int *rank_source, int *rank_dest)
- Returns the ranks of the processes that are a shift of disp steps in coordinate direction
- Useful for nearest neighbor communication in the coordinate directions
  - Use MPI_Cart_coords, MPI_Cart_rank for more general patterns
MPI Graph Topology

- MPI provides routines to specify a general graph virtual topology
  - Graph vertices represent MPI processes (usually one per process)
  - Graph edges indicate important connections (e.g., nontrivial communication between the connected processes)
  - Edge weights provide more information (e.g., amount of communication)
MPI_Dist_graph_create_adjacent

- MPI_Dist_graph_create_adjacent(MPI_Comm oldcomm, int indegree, int sources[], int sourceweights[], int outdegree, int dests[], int destweights[], MPI_Info info, int qreorder, MPI_Comm *newcomm)

- Describe *only* the graph vertex corresponding to the calling process
  - Hence “Dist_graph” – distributed description of graph
- Graph is directed – separate in and out edges
- info allows additional, implementation-specific information
- qreorder if true lets MPI implementation reorder ranks for a better mapping to physical topology
- MPI_UNWEIGHTED may be used for weights arrays
Other Graph Routines

- **MPI_Dist_graph_create**
  - More general, allows multiple graph vertices per process

- **Information on graph**
  - **MPI_Dist_graph_neighbors_count**, **MPI_Dist_graph_neighbors**
Some Results
(Good and Bad)

• A common virtual topology is nearest neighbor in a mesh
  ♦ Matrix computations
  ♦ PDE Simulations on regular computational grids

• Many Large Scale Systems use a mesh as the physical topology
  ♦ IBM Blue Gene series; Cray through XE6/XK7

• Performance can depend on how well the virtual topology is mapped onto the physical topology
Why Mesh Networks?

- **Pros:**
  - Scaling cost of adding a node is constant
  - Nearest neighbor bandwidth proportional to the number of nodes (thus scales perfectly as well)
  - Cabling relatively simple

- **Cons:**
  - Bisection bandwidth does *not* scale with network size
    - For 3D mesh, scales as \( \frac{n^2}{n^3} = \frac{n^{2/3}}{1} \) for \( nxnxn \) mesh
  - Non-nearest neighbor communication suffers from contention
Mesh Performance Limits

• What is the maximum aggregate bandwidth of an n x n x n mesh, assuming:
  ♦ Each interior node sends at bandwidth L to each of its 6 neighbors (±x,±y,±z direction)
  ♦ Edge nodes send to their immediate neighbors

• What is the bisection bandwidth of this network (simple cut along any coordinate plane)?
Mesh Performance

• Aggregate bandwidth
  ♦ Simple, overestimate: $n^3$ nodes * 6 links/node * $L$ bytes/sec/link = $6Ln^3$ bytes/sec
  ♦ More accurate
    • $6L(n-2)^3 + 6(n-2)^25L + 12(n-2)4L + 8(1)3L$
    • i.e., Interior + 6 faces + 12 edges + 8 corners

• Bisection Bandwidth
  ♦ $Ln^2$

• Note: Nearest neighbor bandwidth is more than $n$ times bisection bandwidth

• For $n=24$, $L = 2$GB/sec
  ♦ Neighbor = $L \times 79488 = 159$ TB/sec
  ♦ Bisection = $L \times 576 = 1.2$TB/sec
Communication Cost Includes More than Latency and Bandwidth

- Communication does not happen in isolation
- Effective bandwidth on shared link is $\frac{1}{2}$ point-to-point bandwidth
- Real patterns can involve many more (integer factors)
- Loosely synchronous algorithms ensure communication cost is worst case
Halo Exchange on BG/Q and Cray XE6

- 2048 doubles to each neighbor
- Rate is MB/sec (for all tables)

<table>
<thead>
<tr>
<th></th>
<th>8 Neighbors</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BG/Q</strong></td>
<td></td>
<td><strong>Cray XE6</strong></td>
</tr>
<tr>
<td></td>
<td>Irecv/Send</td>
<td></td>
</tr>
<tr>
<td>World</td>
<td>662</td>
<td>World</td>
</tr>
<tr>
<td>Even/Odd</td>
<td>711</td>
<td>Even/Odd</td>
</tr>
<tr>
<td>1 sender</td>
<td>2873</td>
<td>1 sender</td>
</tr>
</tbody>
</table>

- Rate is MB/sec (for all tables)
Discovering Performance Opportunities

- Lets look at a single process sending to its neighbors.
- Based on our performance model, we expect the rate to be roughly twice that for the halo (since this test is only sending, not sending and receiving)

<table>
<thead>
<tr>
<th>System</th>
<th>4 neighbors</th>
<th>8 Neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Periodic</td>
<td>Periodic</td>
</tr>
<tr>
<td>BG/L</td>
<td>488</td>
<td>490</td>
</tr>
<tr>
<td>BG/P</td>
<td>1139</td>
<td>1136</td>
</tr>
<tr>
<td>BG/Q</td>
<td></td>
<td>2873</td>
</tr>
<tr>
<td>XT3</td>
<td>1005</td>
<td>1007</td>
</tr>
<tr>
<td>XT4</td>
<td>1634</td>
<td>1620</td>
</tr>
<tr>
<td>XE6</td>
<td></td>
<td>5507</td>
</tr>
</tbody>
</table>
Discovering Performance Opportunities

- Ratios of a single sender to all processes sending (in rate)
- *Expect* a factor of roughly 2 (since processes must also receive)

<table>
<thead>
<tr>
<th>System</th>
<th>4 neighbors Periodic</th>
<th>8 Neighbors Periodic</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG/L</td>
<td>2.24</td>
<td>2.01</td>
</tr>
<tr>
<td>BG/P</td>
<td>3.8</td>
<td>2.2</td>
</tr>
<tr>
<td>BG/Q</td>
<td></td>
<td>1.98</td>
</tr>
<tr>
<td>XT3</td>
<td>7.5</td>
<td>8.1</td>
</tr>
<tr>
<td>XT4</td>
<td>10.7</td>
<td>10.7</td>
</tr>
<tr>
<td>XE6</td>
<td></td>
<td>15.6</td>
</tr>
</tbody>
</table>

- BG gives roughly double the halo rate. XTn and XE6 are much higher.
- It should be possible to improve the halo exchange on the XT by scheduling the communication
- Or improving the MPI implementation
Limitations of MPI Process Topology Routines: Cartesian

- **Dims_create**
  - Only for MPI_COMM_WORLD; if strictly implemented, nearly useless
  - Standard defines exact output, makes this a convenience routine for computing factors of an integer. This was the wrong definition

- **Cart routines**
  - Can be implemented, but can be nontrivial in non-mesh network
Limitations of MPI Process Topology Routines: Graph

- Graph routines
  - Complex to implement. No good implementations in general use; research work limited
    - E.g., minimize “bandwidth” in the numerical sparse matrix sense of the connection graph. Does not minimize contention

- One-level
  - Doesn’t address cores/chips, though cart/graph_map could
• MPI-1 and MPI-2 contained a different set of Graph topology routines
  ♦ These required each process to provide the entire graph
  ♦ Simplifies determination of virtual to physical topology mapping
  ♦ Sensible when maximum number of processes was < 200 (when MPI-1 created)
  ♦ These routines are MPI_Graph_xxx
  ♦ Do not use these in new codes
Nonstandard Interfaces

- Many systems provide ways to
  - Control mapping of processes
  - Access the mapping
- Mapping on Job Startup
  - Sometimes called allocation mapping
  - Typically specified by environment variable or command line option
Example: Blue Waters Allocation Mapping

- Environment variable
  - MPICH_RANK_REORDER_METHOD
  - Values:
    - 0 = Round robin by node
    - 1 = Fill each node with processes before going to next node (“SMP ordering”)
    - 2 = Folded by node (0,1,2,...,q,q,q-1,...,0)
    - 3 = Read from file named MPICH_RANK_ORDER

- Mapping to cores within node controlled by –cc and –d options to aprun

- https://bluewaters.ncsa.illinois.edu/topology-considerations
Example Blue Gene/Q Allocation Mapping

• Option to run job:
  ♦ --mapping ABCDET
  ♦ where order of letters indicates which torus coordinate (A-E) or process on node (T) increments (starting from the right)
  ♦ Mapping with a file also possible

• http://www.redbooks.ibm.com/redbooks/pdfs/sg247948.pdf
Mapping at Runtime

• Also known as Rank Reordering
• Create a new communicator that gives each MPI process a new rank to achieve a “better” mapping from virtual to physical topology
  ♦ This is what the MPI Topology routines do
• Requires access to the physical topology
  ♦ No standard method, but many systems provide an API
  ♦ Clusters may provide hwloc

http://www.open-mpi.org/projects/hwloc/
Access to Mesh Topology

• Simple routines available for Blue Waters (Cray systems with Gemini interconnect) and IBM Blue Gene/Q

• Provides access to physical mesh coordinates as well as chip, core number within node

• Example of scalable access to regular network
Access to Mesh Topology

```c
#include <stdio.h>
#include <string.h>
#include "mpi.h"
#include "topoinfo.h"
int main(int argc, char **argv)
{
    topoInfo_t *topoinfo;
    int wrank, verbose=0;
    char leader[10];
    MPI_Init(&argc,&argv);
    if (argv[1] && strcmp(argv[1],"-v") == 0) verbose = 1;
    MPI_Comm_rank(MPI_COMM_WORLD,&wrank);
    snprintf(leader,sizeof(leader),"%d:",wrank);
    topoInit(verbose,&topoinfo);
    topoPrint(stdout,leader,topoinfo);
    topoFinalize(&topoinfo);
    MPI_Finalize();
    return 0;
}
```
Impact of Other Jobs

• Even with a perfect mapping, programs can suffer from interference with other jobs
• Can be reduced by topology-aware scheduling
• Layout of I/O nodes, adaptive routing can create contention even with topology-aware scheduling
• In this example, either the blue job or the pink job can communicate without contention, but together they share all of the “x” links in the pink job
Readings

- Generic Topology Mapping Strategies for Large-scale Parallel Architectures, Hoefler and Snir
  http://dx.doi.org/10.1145/1995896.1995909

- Implementing the MPI Process Topology Mechanism, Traeff
  http://www.computer.org/csdl/proceedings/sc/2002/1524/00/15240028-abs.html

- Avoiding Hot Spots on Two-Level Direct Networks, Bhatelte, Jain, Gropp, Kale
  http://dl.acm.org/citation.cfm?doid=2063384.2063486