Principles of High Performance Computing (ICS 632)

Virtual Topologies for

Distributed Memory Computing

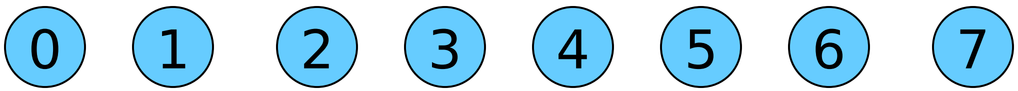
Beyond MPI_Comm_rank()?

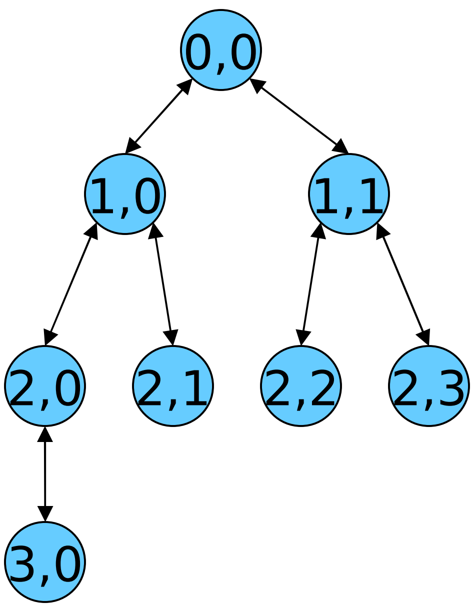
- So far, MPI gives us a unique number for each processor
- With this one can do anything
- But it’s pretty inconvenient because one can do anything with it
- Typically, one likes to impose constraints about which processor/process can talk to which other processor/process
- With this constraint, one can then think of the algorithm in simpler terms
  - There are fewer options for communications between processors
  - So there are fewer choices to implementing an algorithm

Virtual Topologies?

- MPI provides an abstraction over physical computers
  - Each host has an IP address
  - MPI hides this address with a convenient numbers
  - There could be multiple such numbers mapped to the same IP address
  - All “numbers” can talk to each other
- A Virtual Topology provides an abstraction over MPI
  - Each process has a number, which may be different from the MPI number
  - There are rules about which “numbers” a “number” can talk to
- A virtual topology is defined by specifying the neighbors of each process

Implementing a Virtual Topology

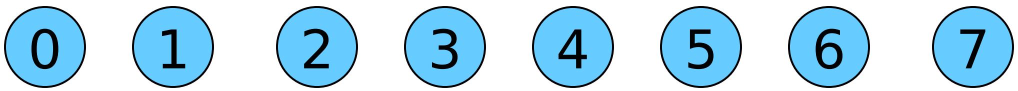


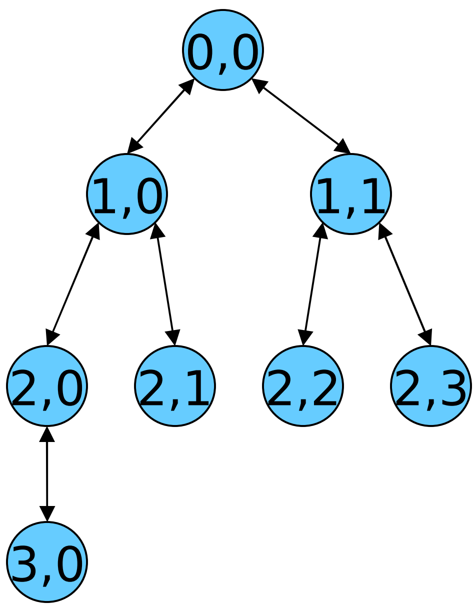


$(i,j) = (\text{floor}(\log2(\text{rank}+1)), \text{rank} - 2^{\max(i,0)}+1)$

$\text{rank} = j - 1 + 2^{\max(i,0)}$

Implementing a Virtual Topology





$(i,j) = (\text{floor}(\log 2(\text{rank}+1)), \text{rank} - 2^{\max(i,0)}+1)$

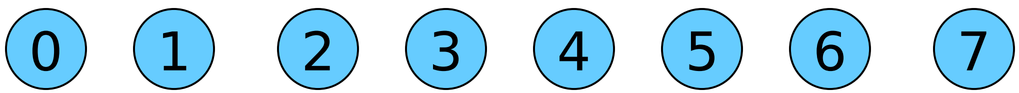
$\text{rank} = j - 1 + 2^{\max(i,0)}$

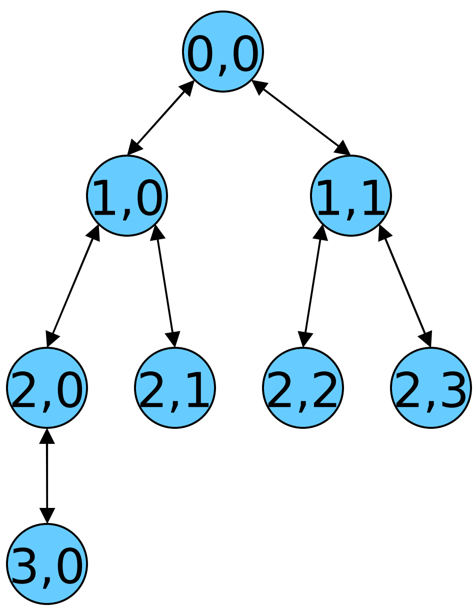
my_parent(i,j) = (i-1, floor(j/2))

my_left_child(i,j) = (i+1, j*2), if any

my_right_child(i,j) = (i+1, j*2+1), if any

Implementing a Virtual Topology





(i,j) = (floor(log2(rank+1)), rank - $2^{max(i,0)}$+1)

rank = j -1 + $2^{max(i,0)}$

my_parent(i,j) = (i-1, floor(j/2))

my_left_child(i,j) = (i+1, j*2), if any

my_right_child(i,j) = (i+1, j*2+1), if any

MPI_Send(…, rank(my_parent(i,j)), …)

MPI_Recv(…, rank(my_left_child(i,j)), …)

Typical Topologies

- Common Topologies (see Section 3.1.2)
  - Linear Array
  - Ring
  - 2-D grid
  - 2-D torus
  - One-level Tree
  - Fully connected graph
  - Arbitrary graph
- Two options for all topologies:
  - Monodirectional links: more constrained but simpler
  - Bidirectional links: less constrained but potential more complicated
    - By “complicated” we typically mean more bug-prone
- We’ll look at Ring and Grid in detail

Main Assumption and Big Question

- The main assumption is that once we've defined the virtual topology we forget it's virtual and write parallel algorithms assuming it's physical
  - We assume communications on different (virtual) links do not interfere with each other
  - We assume that computations on different (virtual) processors do not interfere with each other
- The big question: How well do these assumptions hold?
  - The question being mostly about the network
- Two possible "bad" cases
- Case #1: the assumptions do not hold and there are interferences
  - We'll most likely achieve bad performance
  - Our performance models will be broken and reasoning about performance improvements will be difficult
- Case #2: the assumptions do hold but we leave a lot of the network resources unutilized
  - We could perhaps do better with another virtual topology

Which Virtual Topology to Pick

- We will see that some topologies are really well suited to certain algorithms
- The question is whether they are well-suite to the underlying architecture
- The goal is to strike a good compromise
  - Not too bad given the algorithm
  - Not too bad given the platform
- Fortunately, many platforms these days use switches, which support naturally many virtual topologies
  - Because they support concurrent communications between disjoint pairs of processors
- As part of a programming assignment, you will explore whether some virtual topology makes sense on our cluster

Topologies and Data Distribution

- One of the common steps when writing a parallel algorithm is to distribute some data (array, data structure, etc.) among the processors in the topology
  - Typically, one does data distribution in a way that matches the topology
  - E.g., if the data is 3-D, then it’s nice to have a 3-D virtual topology
- One question that arises then is: how is the data distributed across the topology?
- In the next set of slides we look at our first topology: a ring