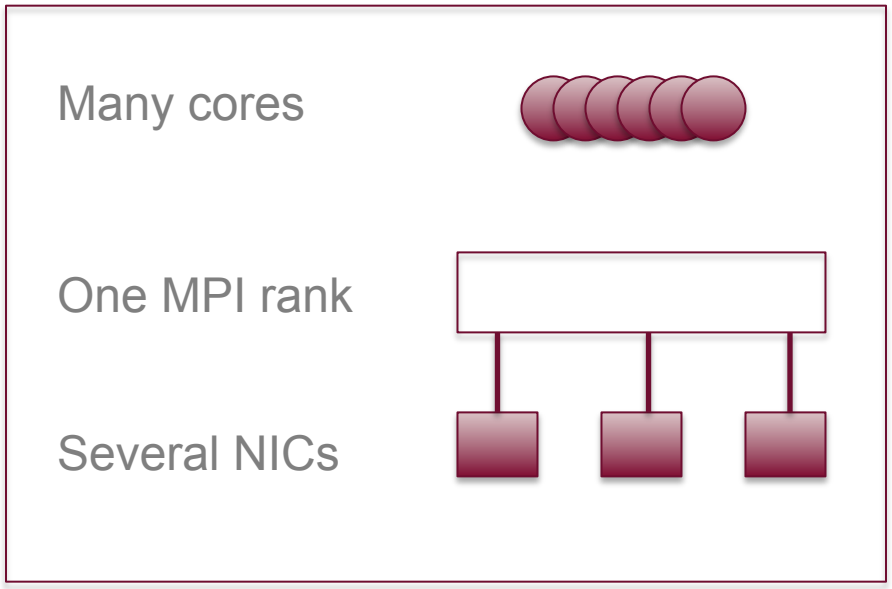
MPI Processes and Shared Memory

Assumptions

- Number of threads per node will increase dramatically
- Nodes will have multiple NICs
- Intranode communication will use shared memory (MPI+X)
  - May not be coherent and will be very NUMA

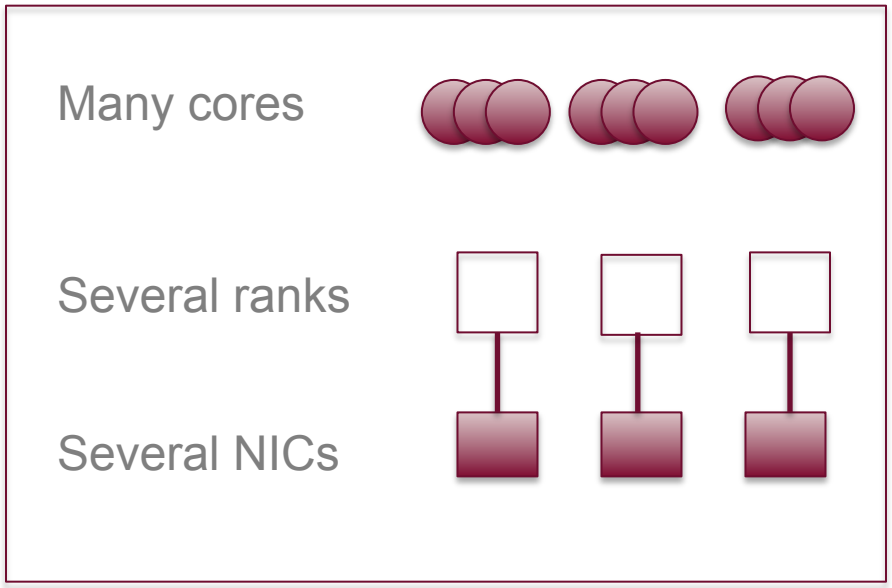
Solution A -- One MPI Rank per Node



Problems:

- Hard to get high message rate: MPI semantics essentially force sequential matching of incoming messages to pending receives
- Hard to implement if memory is not coherent
- Will perform badly in a NUMA system

Solution B -- Multiple MPI Processes per Node, Each With a Fixed Thread Pool



Problem:

- Assumes (parent) threads are statically associated with MPI endpoints; does not fit OpenMP and other dynamic programming models

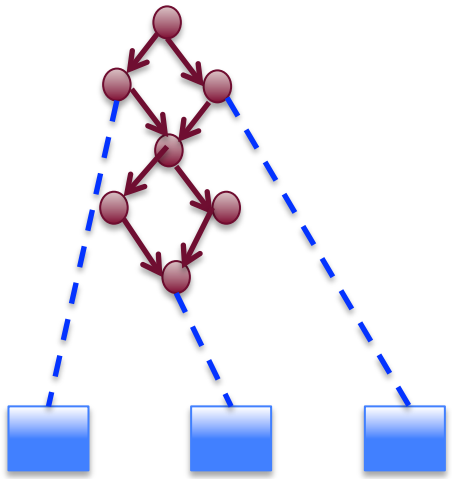
Hybrid Programming Models

- MPI+OpenMP

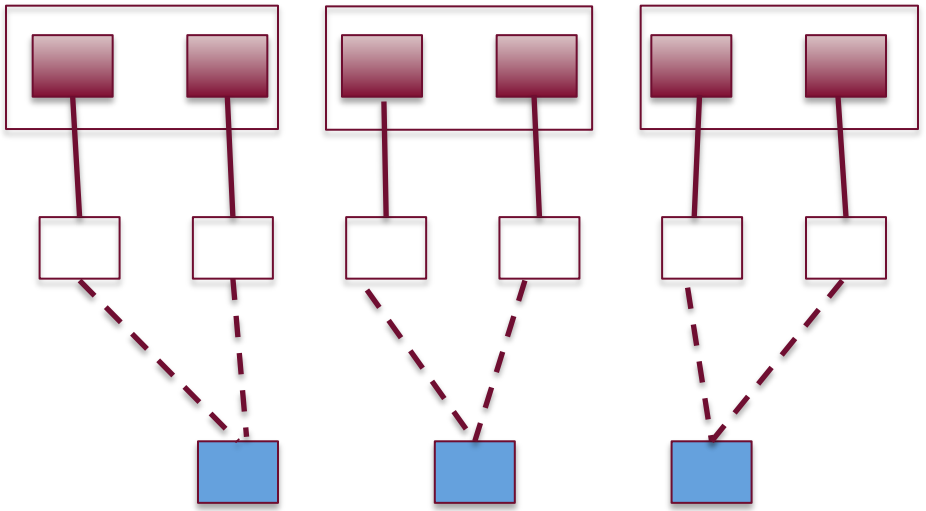
Dynamically spawned "threads"

Dynamic associations

Several ranks



- MPI+PGAS
  - Fixed number of "threads"
  - Multiple threads per address space
  - One MPI endpoint per thread



Remarks

- MPI never specified that an “MPI process” runs in a separate address space
  - Different implementations have equated “MPI processes” with OS threads or run-time tasks

- Need to define MPI semantics with
  - as few assumptions as possible about what an “MPI process” is
  - as few assumptions as possible about what a “thread” is
  - No assumptions about shared memory mechanisms

- Very desirable to support thread migration across MPI processes

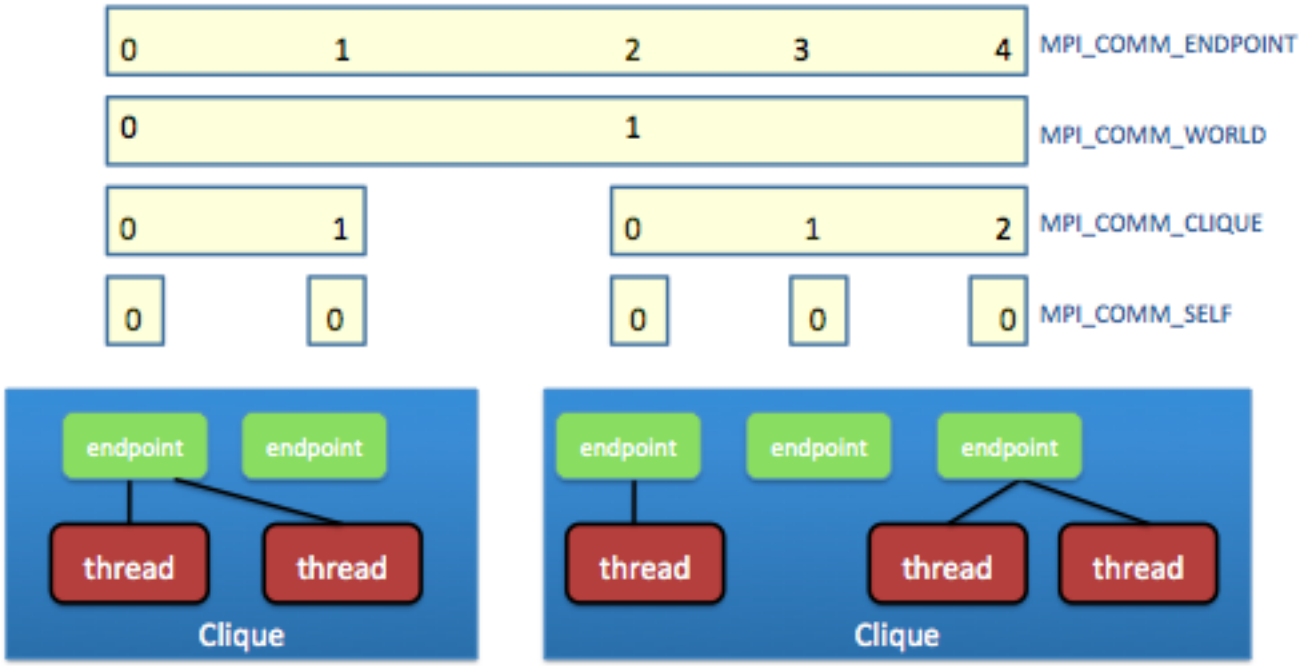
Solution C: Multiple MPI Processes per Node, Threads can Join MPI Processes

Definitions

- *MPI Endpoint*: set of resources that enable MPI communications; correspond to an MPI rank
- *Thread*: unit of execution
  - A thread attached to an endpoint can use this endpoint to make MPI calls (using the corresponding ranks)
- *MPI Process*: an MPI endpoint with threads attached to it
- *MPI Clique*: set of endpoints that can be accessed by same threads (i.e., a thread can migrate from one endpoint to ny other within the same clique)
  - Current model: Clique size = 1
  - OpenMP model: Clique = all endpoints attached to an OpenMP program
  - Other models are possible

Initialization

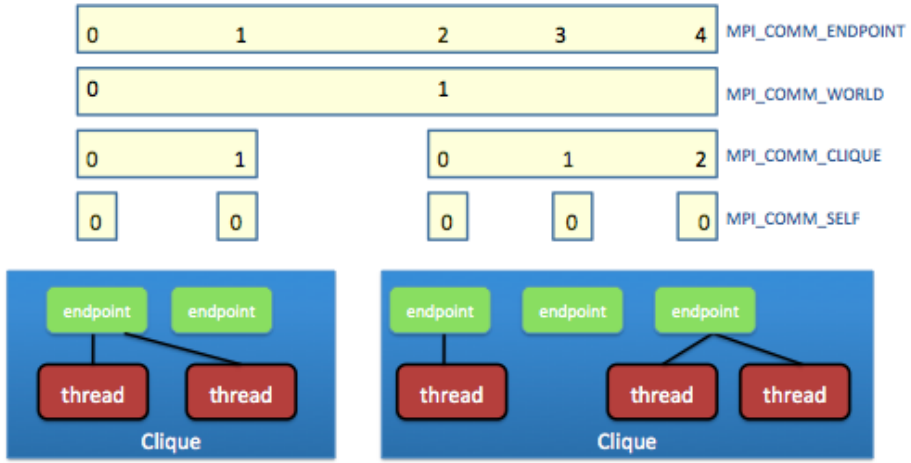
- Set of endpoints, their mapping to nodes, and their partitioning into cliques determined externally (e.g., by mpiexec)
- 4 communicators available after initialization



Communicators

- _SELF provided for backward compatibility
- _ENDPOINTS needed (to include all endpoints)
- _CLIQUE defines the clique (could be avoided using MPI_COMM_SPLIT_TYPE)
- _WORLD provided for backward compatibility

- Computation starts with one thread per clique
- This thread is attached to first endpoint in clique – the endpoint in _WORLD
- Children threads are attached to same endpoint as parent
  - No change in programs using one endpoint per clique!



One New Command

- MPI_THREAD_ATTACH(rank, comm)
  - Thread detaches fro current endpoint and attaches to new endpoint
  - Error raised if new endpoint is not in same clique

- (Almost) no other changes
  - New info for MPI_COMM_SPAWN
- INIT and FINALIZE done (at least) once per clique

Implementation

- When thread makes MPI call need to find which endpoint it is attached to
- *Current*: information is static
  - Found at a fixed address
- *New*: Information can change
  - Found at a thread private location
  - May require an additional level of indirection

- No other changes are required