Shared Memory and Distributed Memory & Interconnection Networks

Module 2.5

Peter J. Hawrylak

Module Learning Objectives

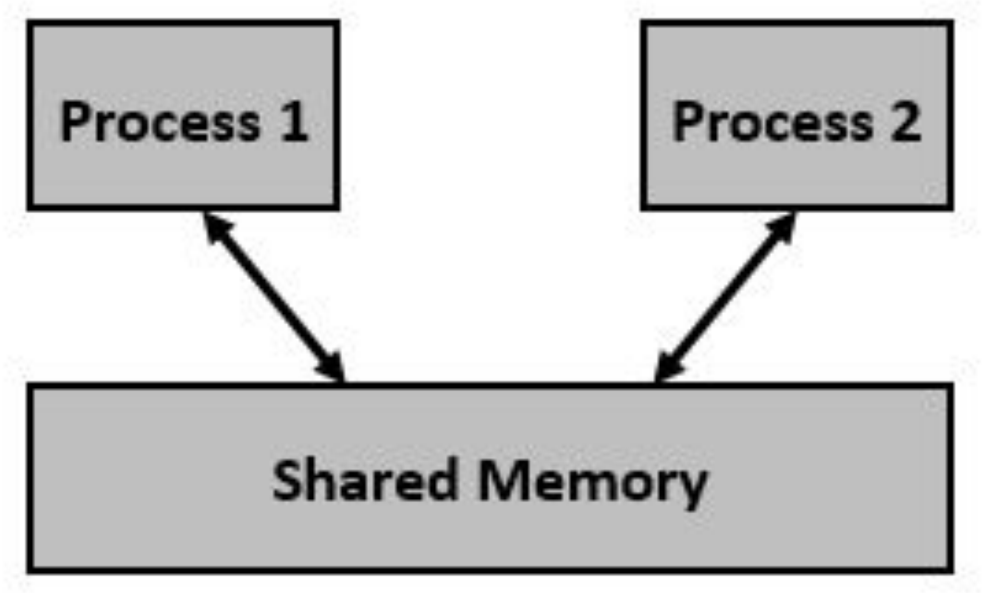
- Describe the difference between a shared memory system and a distributed memory system.
- Categorize common parallel programming tools, including OpenMP, MPI, and PThreads, as either shared memory or distributed memory approaches.
- Define key concepts in distributed memory systems, including bisection bandwidth and redundancy
- Estimate the number of links needed in a distributed memory system with a given architecture.
- Estimate the number of “hops” a message needs in a distributed memory system with a given architecture.

Shared and Distributed Memory

- Important to know **where** data are and **how** they are distributed.
  - Are the data all in the same processing element? □ If YES then it is shared memory.
  - Are the data distributed over multiple processing elements? □ IF YES then it is distributed memory.
- Shared Memory languages
  - OpenMP and pThreads
- Distributed Memory languages
  - MPI
- Most real-world problems employ a combination of shared and distributed memory

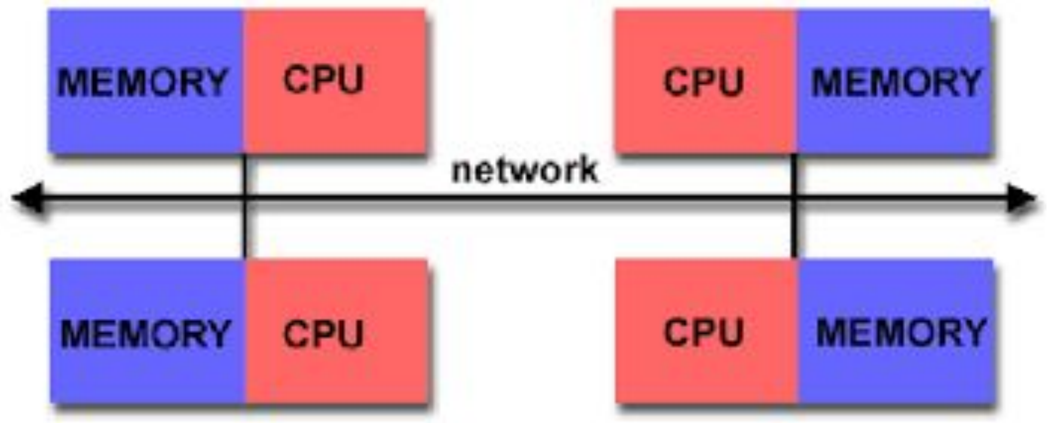
Shared Memory

- All processes have access to all of the data and that data is physically located with the processing element.
- Shared Memory Benefits
  - Uniform memory access – same time to access data for ALL processes
  - Easy to share data between processes – ALL processes can update ALL data items
- Shared Memory Drawbacks
  - Does not scale well – limited to HOW MANY processes can run on a single processing element



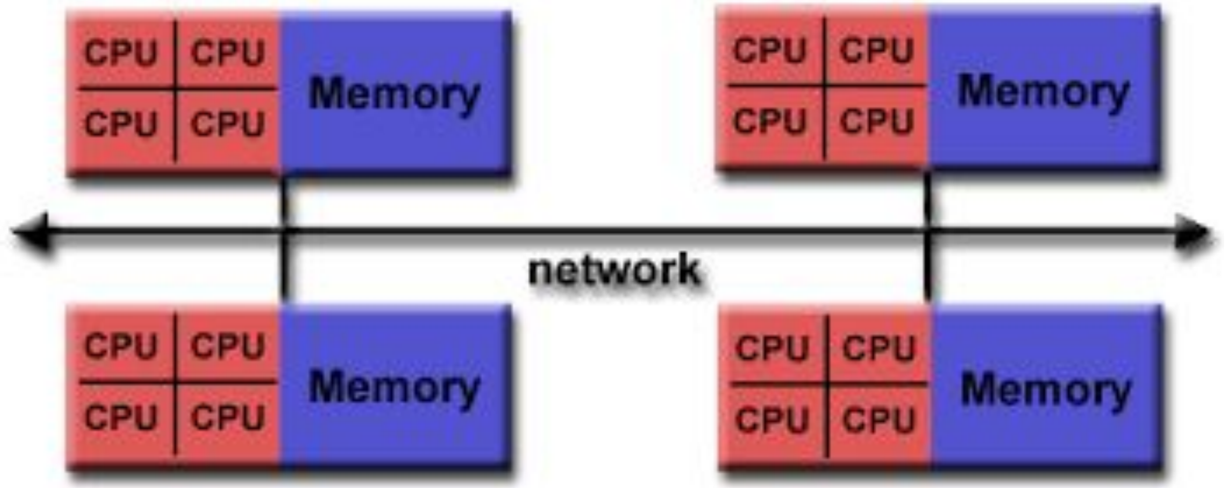
Distributed Memory

- All processes can access ALL data, but that data MAY or MAY NOT be located on the same physical system that is hosting the process.
- Distributed Memory Benefits
  - Large memory – combine many memories together
  - Scalability – more physical nodes = more processes
- Distributed Memory Drawbacks
  - Non-Uniform memory access - time to access data varies depending where data are located.
  - More complicated – may need to manage data location and sharing of commonly used data among physical nodes.



Hybrid Model

- Combine shared and distributed memory.
- Can combine languages too
  - They are built to work together
  - Shared Memory languages for SINGLE processing element
  - Distributed Memory languages for scaling to multiple physical nodes
- Allows maximum scalability
  - Use ALL of your resources efficiently

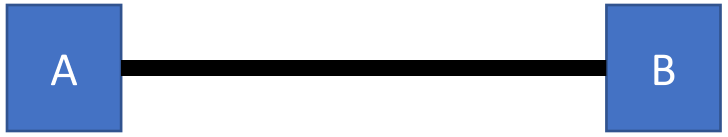


Should I Use Shared or Distributed Memory?

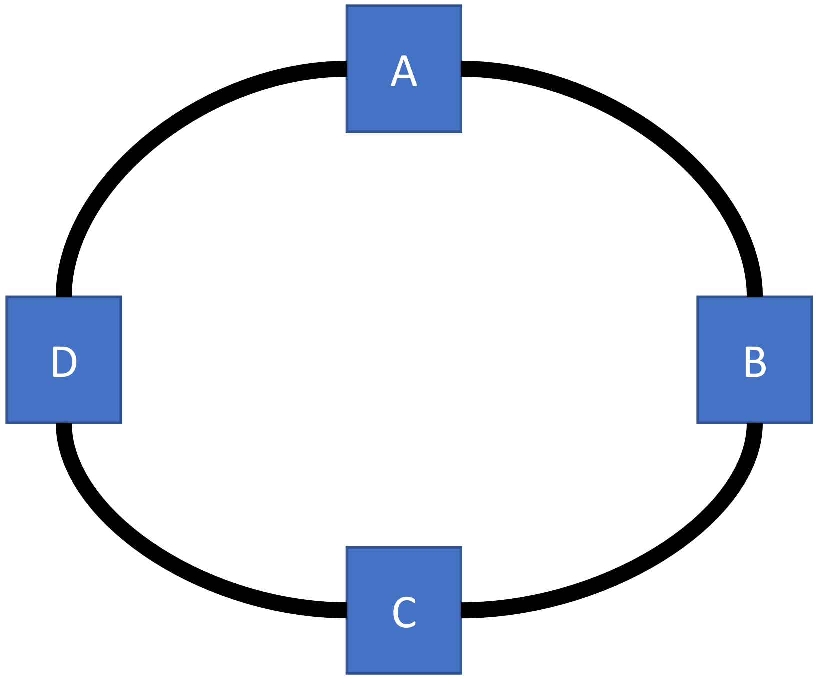
- Is there a lot of updating of the same data item?
  - Can you break this data structure into smaller parts that could be operated on independently? □ Do this if you select distributed memory
  - If not the shared memory is a good choice.
- Is there a need to scale beyond a single physical node?
  - If yes then you will NEED to use distributed memory.

Connecting Physical Nodes Together (1)

Point-to-Point Connection



Ring Topology



All connections are assumed to be bi-directional unless noted with an arrow at one end.

Connecting Physical Nodes Together (2)

- These need more complicated traffic management
  - Switches and routers
- Toroidal Mesh
  - Donut shape
- Hypercube
- Crossbar
  - Grid (north, south, east, and west) of interconnects
  - Messages may need to go through the network communication is NOT point-to-point
- Fully Connected Network
  - Each physical nodes a DIRECT connection to every other physical node
  - Requires n^2 links
  - Very expensive and not widely used above 4-8 physical nodes

Communication Between Physical Nodes

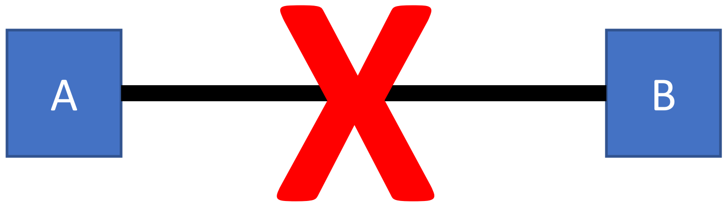
- Communication is SLOW
  - Both within physical node and between physical nodes
- Number of hops varies depending on topology
- Topology impacts congestion on network
  - Minimize communication bottlenecks network theory/design
- Generally, the more connections the fewer hops and fewer bottlenecks
  - Links cost money and space (real-estate) in the computing center

Handling Congestion and Link Failure

- Approaches to congestion
  - Increase speed and through-put to reduce bottlenecks
  - Increase number of links to by-pass to bottlenecks
  - Increase number of links to ”add more lanes” to the bottleneck area
- Approaches to link failure
  - Multiple routes allow alternative paths (detours)
- Bisection bandwidth – number of links that must fail to disconnect a network
  - Gives a hint at how congested a network can become
  - Low value means much of the traffic is passing through a few links

Bisection Bandwidth

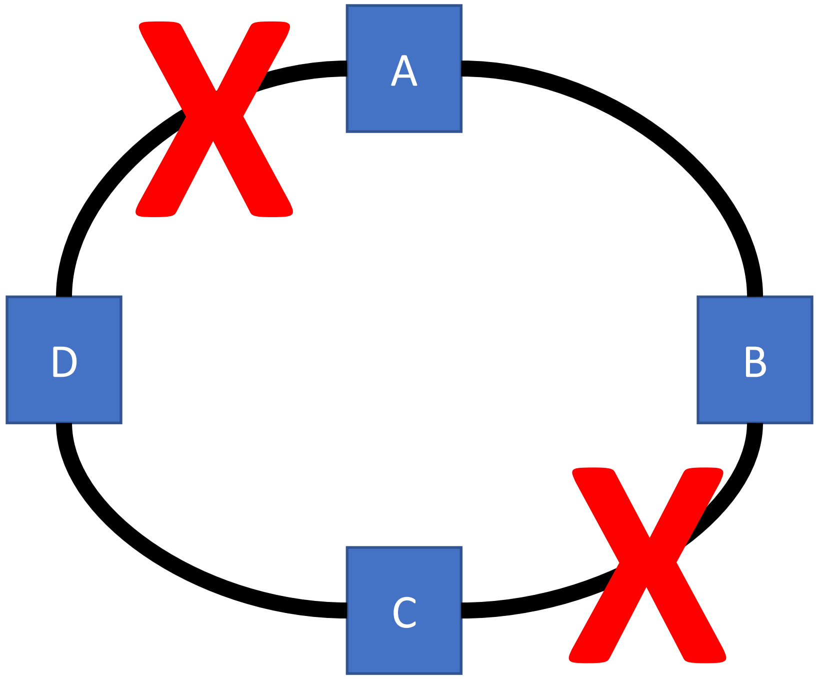
Point-to-Point Connection



Bisection bandwidth = 1

Ring Topology

Bisection bandwidth = 2



All connections are assumed to be bi-directional unless noted with an arrow at one end.

Summary

- Shared Memory – All memory on the same physical processing element, Uniform Memory Access time
  - Example Parallel Programming Languages: OpenMP and PThreads
- Distributed Memory – Memory is dispersed among physical processing elements, Non-Uniform Memory Access time
  - Example Parallel Programming Languages: MPI
  - How many hops to get data in a distributed memory system?
  - Bisection Bandwidth – How many links must fail to disconnect part of the network.
  - Redundancy – Improves bisection bandwidth AND message throughput
  - Different network topologies require different numbers of links but provide different levels of service (e.g., message throughput, message latency, redundancy)

Thank You
Questions?