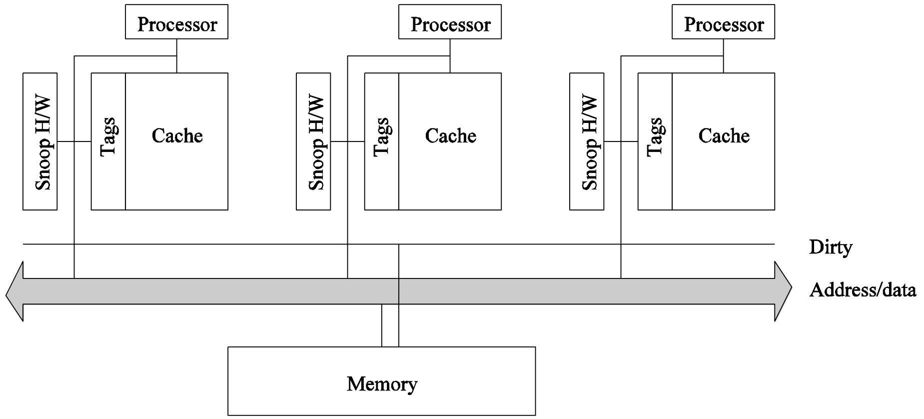
CS5375 Computer Systems Organization and Architecture

Lecture 21

Instructor: Yong Chen, Ph.D.
Department of Computer Science
Texas Tech University
Yong.Chen@ttu.edu, 806-834-0284

Review of Last Lecture

- Thread-Level Parallelism
  - Targeted for tightly-coupled shared-memory multiprocessors
  - Symmetric multiprocessors
  - Distributed shared memory
    - Non-uniform memory access/latency (NUMA)
  - **Cache coherence**
    - Processors may see different values through their caches (private caches)
  - Enforcing coherence: **update protocol** v.s. **invalidate protocol**
    - Examples of invalidate protocol
  - Snoopy/snooping cache systems
    - Handel how invalidates are sent to the right processors
    - A broadcast media listens to all invalidates and performs coherence operations



Outline

- Thread-Level Parallelism
  - Directory Based Cache Coherence Systems
  - Programming Models

- Distributed Memory, Large-scale Computers

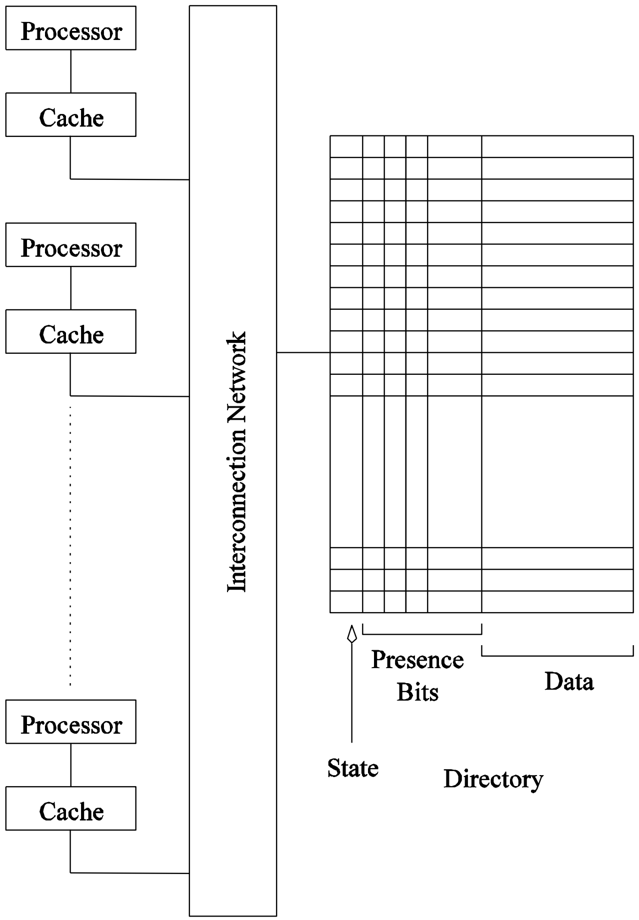
Directory Based Cache Coherence Systems

- In snoopy caches, each coherence operation is sent to all processors
  - This is an inherent limitation

- Why not send coherence requests to only those processors that need to be notified?

- This is done using a directory, which maintains a presence vector for each data item (cache line) along with its state

Directory Based Cache Coherence Systems



Architecture of typical directory based systems: a centralized directory

Performance of Directory Based Cache Coherence

- The need for a snoopy hardware is replaced by the directory

- The additional bits to store the directory may add significant overhead
  - How much overhead?

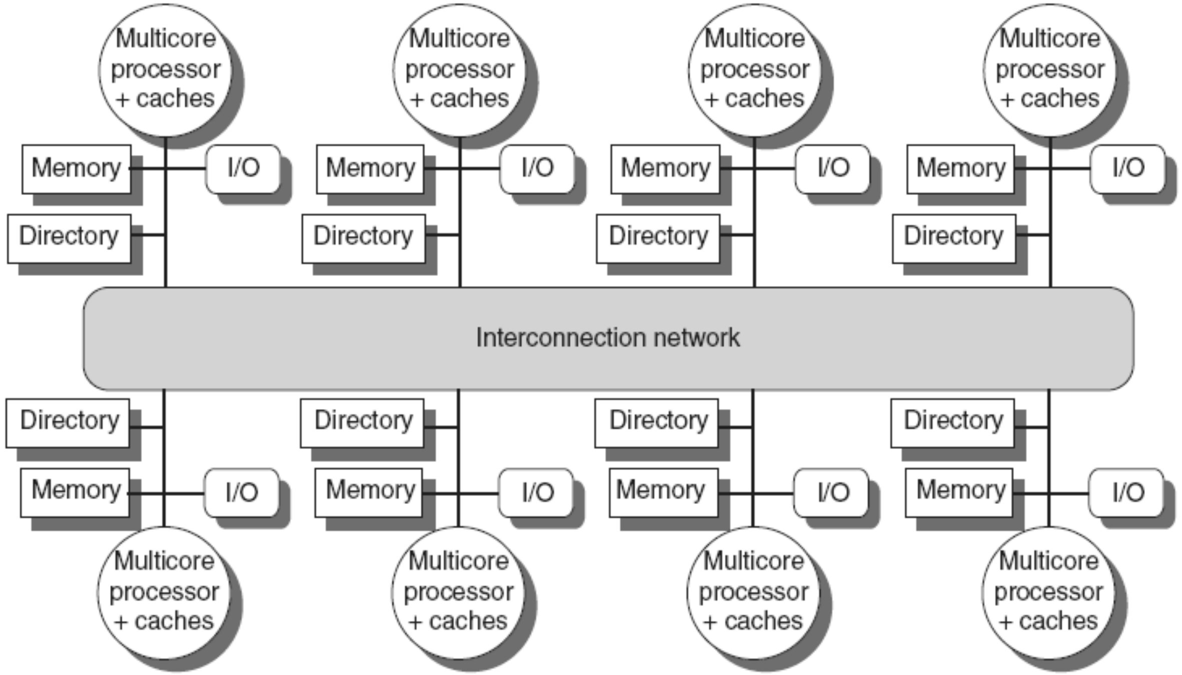
  $p \times n$, where *p* is the number of processors and *n* is the number of total cache lines (cache blocks) of *p* processors

- The interconnection network must be able to carry all the coherence requests

- The directory is a point of contention, therefore, distributed directory schemes can be used

Directory Based Cache Coherence Systems

- Permit $O(p)$ simultaneous coherence operations
- More scalable than snoopy or centralized directory systems
- Remains significant overhead of directory storage



Architecture of typical directory based systems: a distributed directory

Programming Models

- Parallel programming models (any) focus on providing support for expressing (via primitives/API):

- Concurrency
  - Create multiple processes/threads

- Synchronization
  - Coordinate these processes/threads to ensure correct results

Programming Models (cont.)

- A *thread* is a single stream of control in the flow of a program.

- A program like:

```
for (row = 0; row < n; row++)
 for (column = 0; column < n; column++)
     c[row][column] =
     dot_product( get_row(a, row),
                  get_col(b, col));
```

can be transformed to:

```
for (row = 0; row < n; row++)
 for (column = 0; column < n; column++)
     c[row][column] =
         create_thread( dot_product(get_row(a, row),
 get_col(b, col)));
```

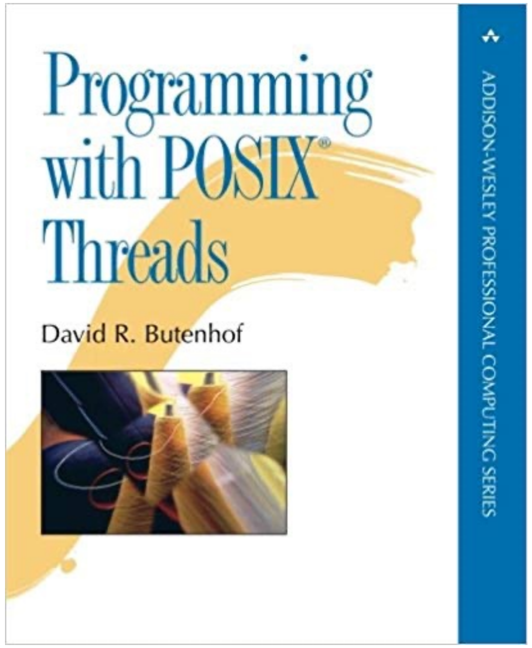
Note that there're no any dependences among dot_product() thus we can create multiple threads to solve them concurrently.

The POSIX Thread API

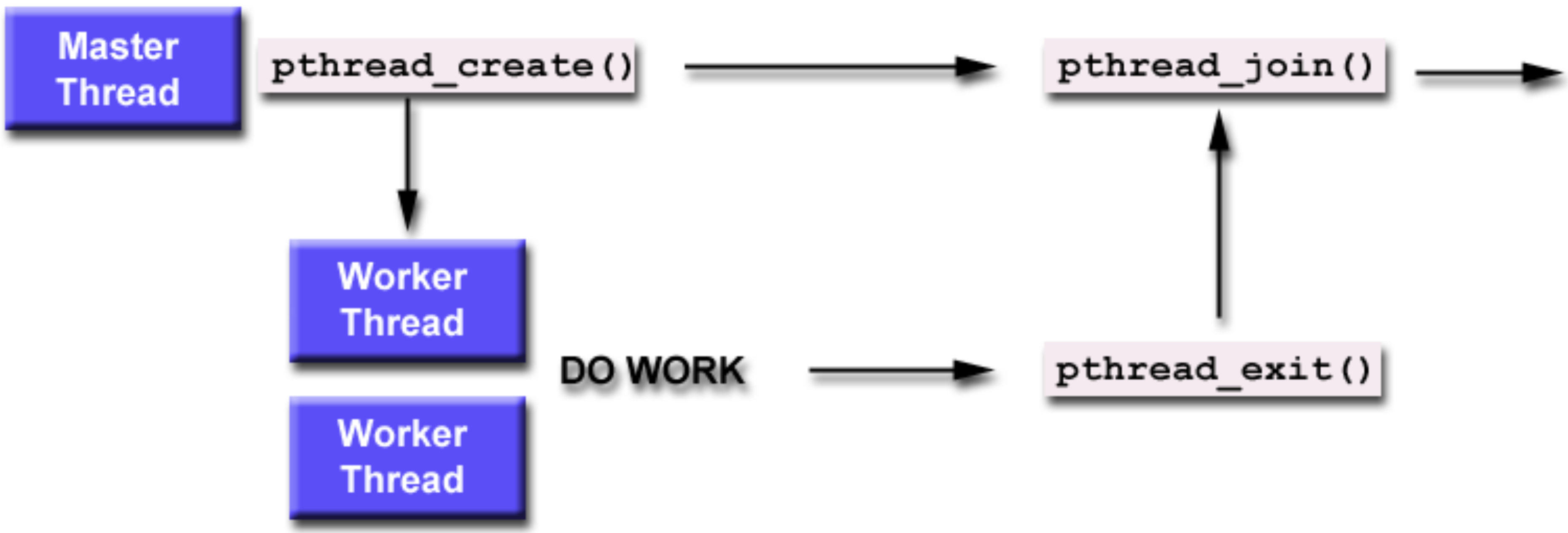
- Commonly referred to as Pthreads, POSIX has emerged as the standard threads API, supported by most vendors
  - IEEE standard 1003.1c-1995 POSIX API
  - 2018 edition: https://pubs.opengroup.org/onlinepubs/9699919799/nframe.html

- Concepts are largely independent of the API
  - Similar to other programming models with other thread APIs

- “Programming with POSIX Threads”, by David R. Butenhof, ISBN-10: 0201633922, ISBN-13: 978-0201633924



Pthreads Programming Model



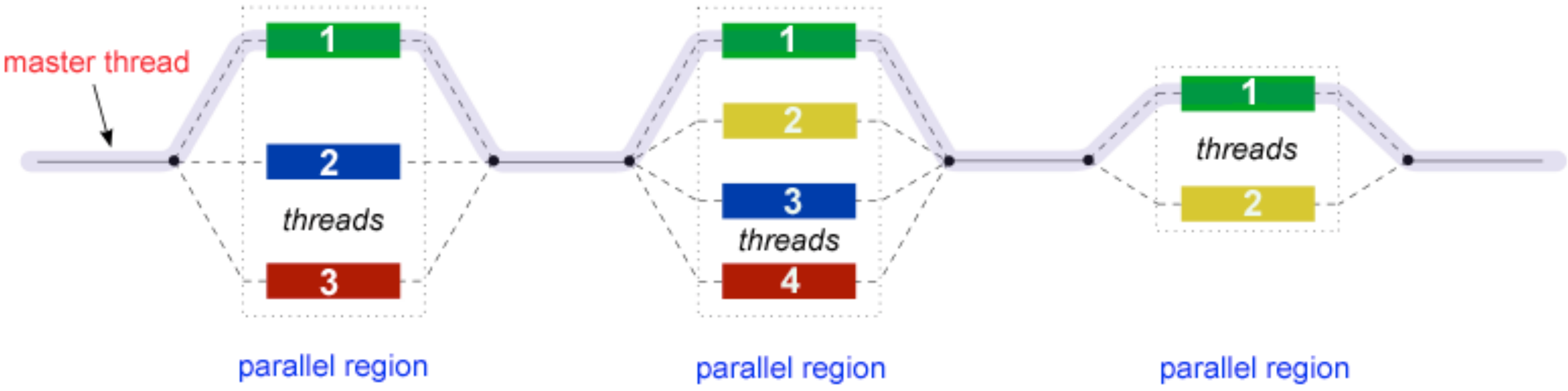
Fork-join model of parallel execution

OpenMP: A Standard Directive Based Programming Model

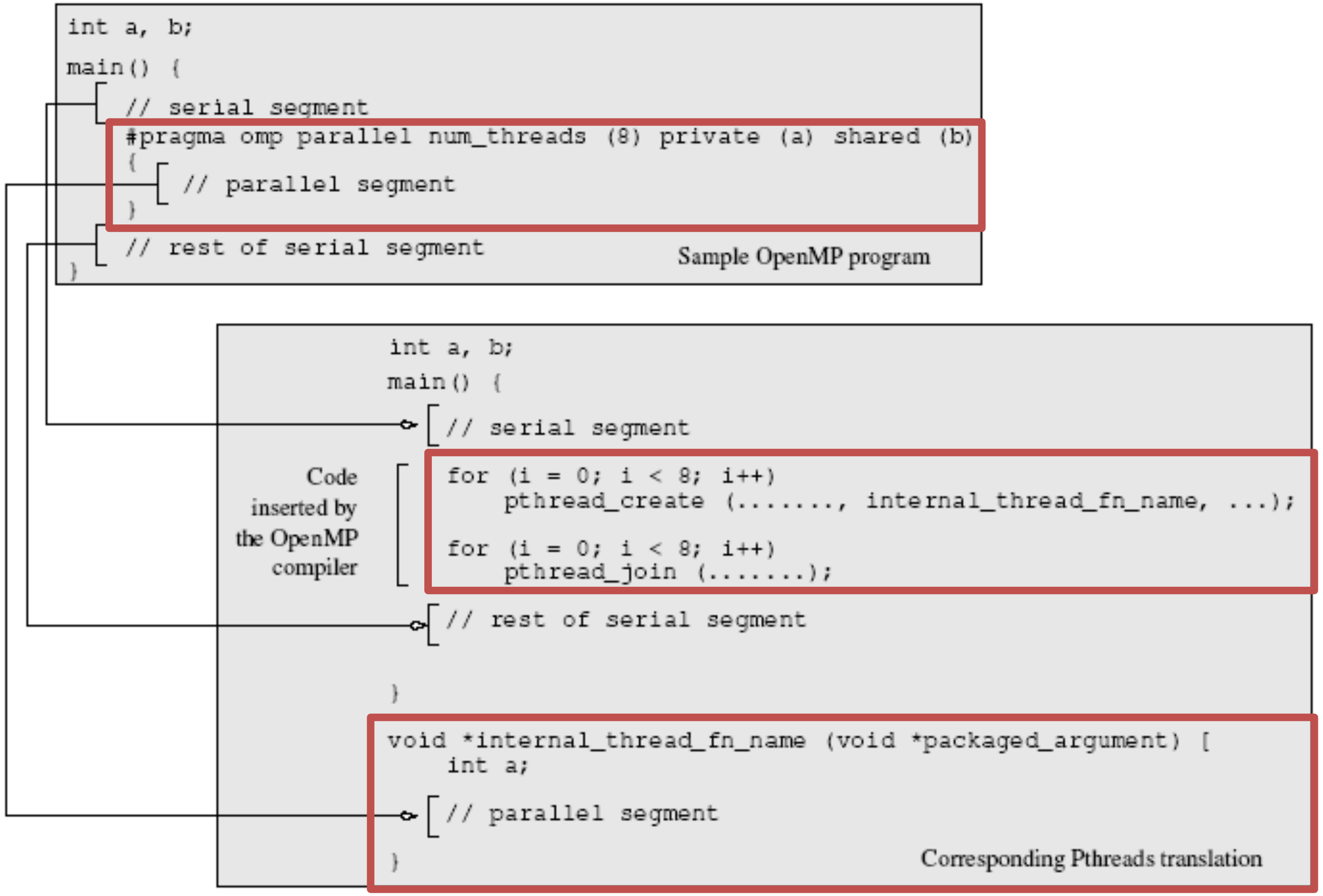
- Pthreads still need explicit management of threads, low-level
- High-level constructs/directives desired
- OpenMP provides a directive-based API that can be used with FORTRAN, C, and C++ for programming shared address space machines.
  - OpenMP: Open Multi-Processing
- OpenMP directives provide support for concurrency, synchronization, and data handling while avoiding the need for explicitly setting up threads, distributing tasks, managing mutex locks, etc.

OpenMP Programming Model

- OpenMP also uses the fork-join model of parallel execution
  - Begin as a single thread: the master thread and executes sequentially until the first parallel region construct is encountered
  - Fork: the master thread then creates a team of parallel threads
    - The statements in the program that are enclosed by the parallel region construct are then executed in parallel among the various team threads.
  - Join: when the team threads complete, they synchronize and terminate, leaving only the master thread



OpenMP V.S. Pthreads Programming Model



- A sample OpenMP program along with its Pthreads translation that might be performed by an OpenMP compiler

OpenMP Standard

- OpenMP standard website: https://www.openmp.org/
  - API specifications, FAQ, presentations, discussions, etc.

- OpenMP specifications
  - https://www.openmp.org/specifications/

- Book: “Using OpenMP: Portable Shared Memory Parallel Programming”

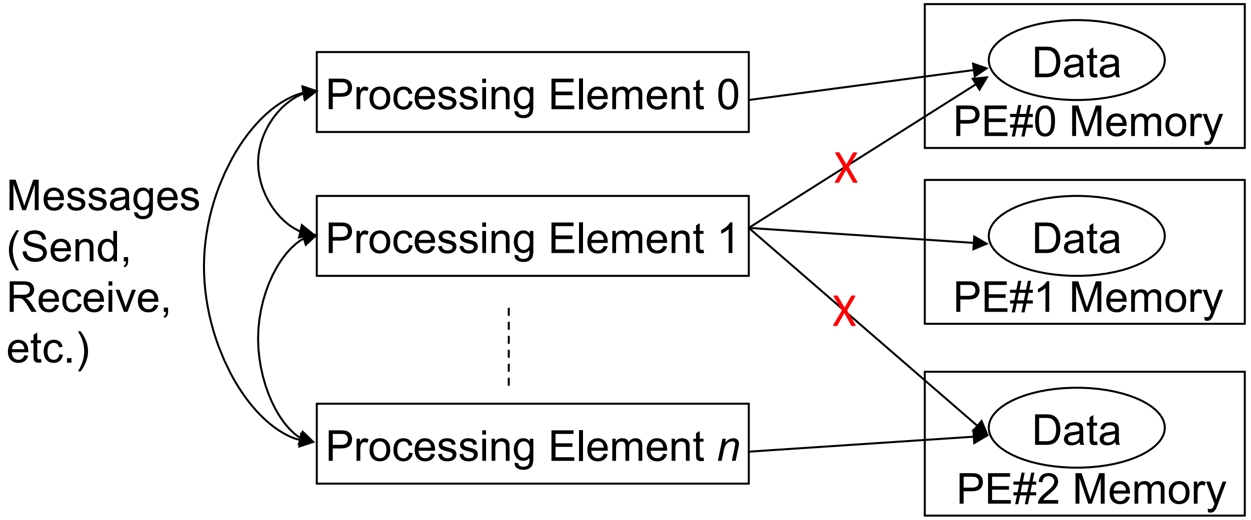
Outline

- Thread-Level Parallelism
  - Directory Based Cache Coherence Systems
  - Programming Models

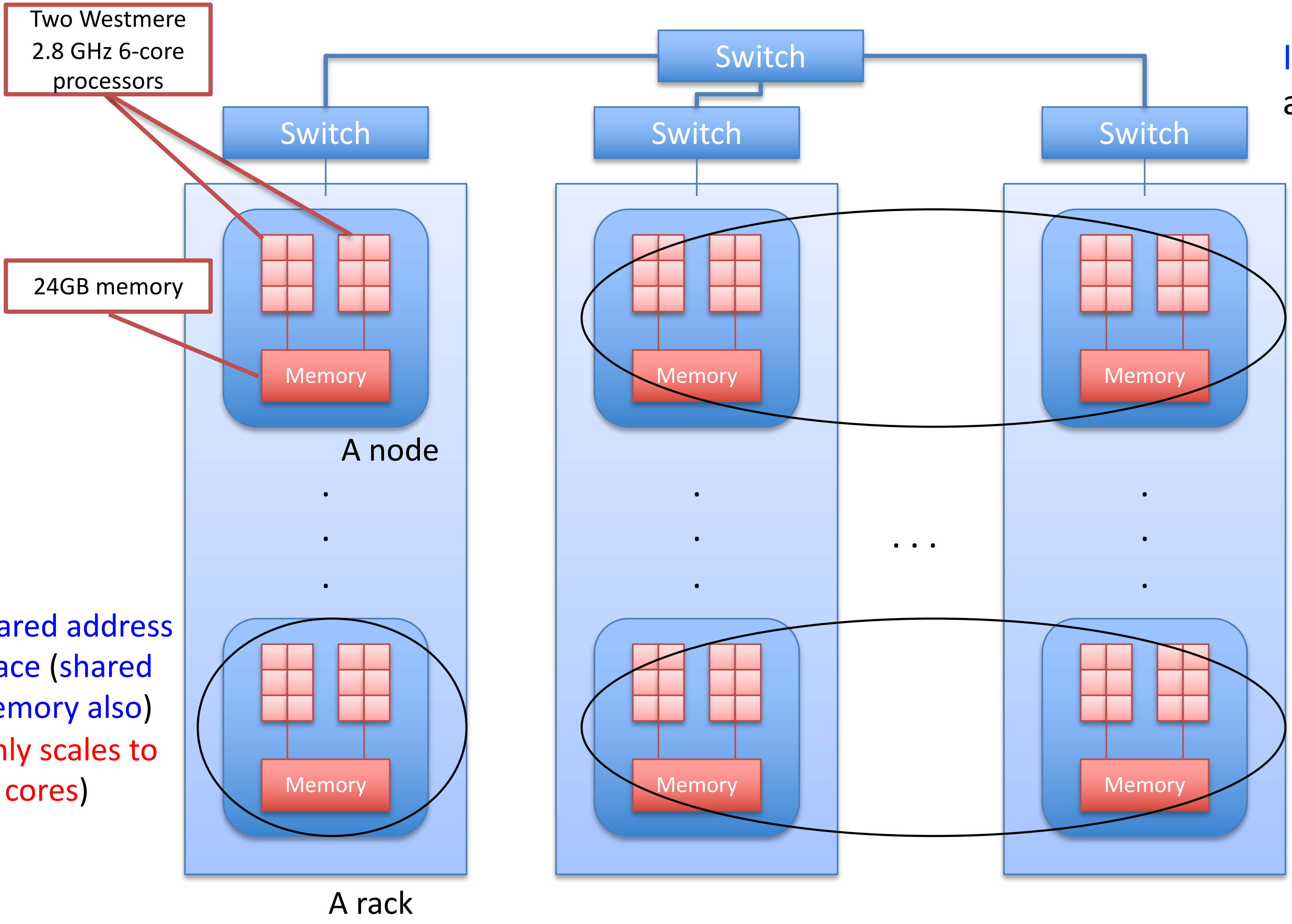
- Distributed Memory, Large-scale Computers

Distributed Memory Architectures

- "Shared nothing:" each processor (or PE, processing element) has a private address space
- Processors/PEs can only directly access local data
  - There's no shared, global memory (or no shared, global address space to be more precise)
- Interaction?
  - These platforms are programmed using (variants of) message passing primitives, e.g. send and receive
  - Libraries such as MPI (message-passing interface), sockets, MapReduce provide such primitives



A Hrothgar@TTU Parallel Computer Example



Interconnect (topology also matters)

Potentially shared address space (specially NUMA) if with HW/SW support (note distributed memory!)

Shared address space (shared memory also) (only scales to 12 cores)

Distributed address space (distributed memory also) (can scale to 12*640=7680 cores)

Shared v.s. Distributed Address Space

- Shared address space: pros and cons
  - Global address space view for programmers, easier to program usually
  - Implicit communication
  - Need hardware/software to support view, complexity in system design
  - Not easy to be scalable

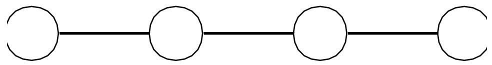
- Distributed address space: pros and cons
  - Requires little hardware support, other than an interconnection network
  - Easy to scale up
  - No global address space view, more difficult to program
  - Need explicit communication

Multiprocessors v.s. Multicomputers

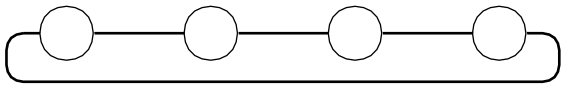
- "Multiprocessors": platforms that provide shared-address-space
- "Multicomputers": platforms that do not provide shared-address space and need message passing communications

Interconnect Topology: Linear Array

- In a linear array, each node has two neighbors, one to its left and one to its right
- If the nodes at either end are connected, we refer to it as a ring or 1-D torus



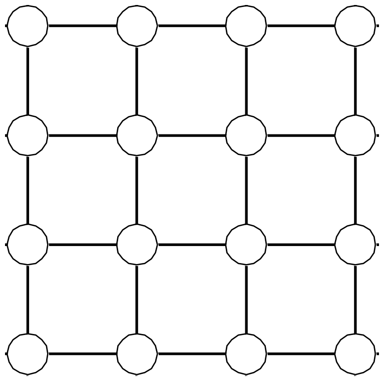
(a) Linear array: with no wraparound links



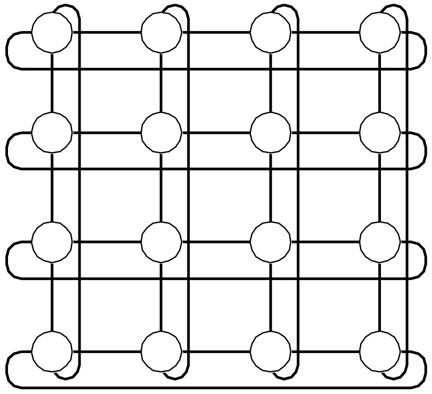
(b) Ring or 1-D torus: with wraparound link.

Interconnect Topology: Meshes

- A generalization of linear array to 2 dimensions has nodes with 4 neighbors, to the north, south, east, and west



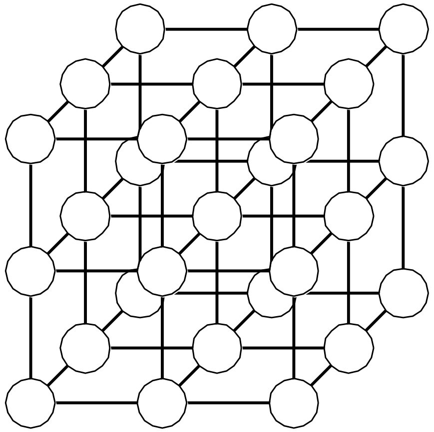
(a) 2-D mesh: without wraparound links



(b) 2-D torus: with wraparound links

Interconnect Topology: Generalized Meshes

- A further generalization to *d* dimensions has nodes with *2d* neighbors (except nodes on the periphery)
- 3-D mesh

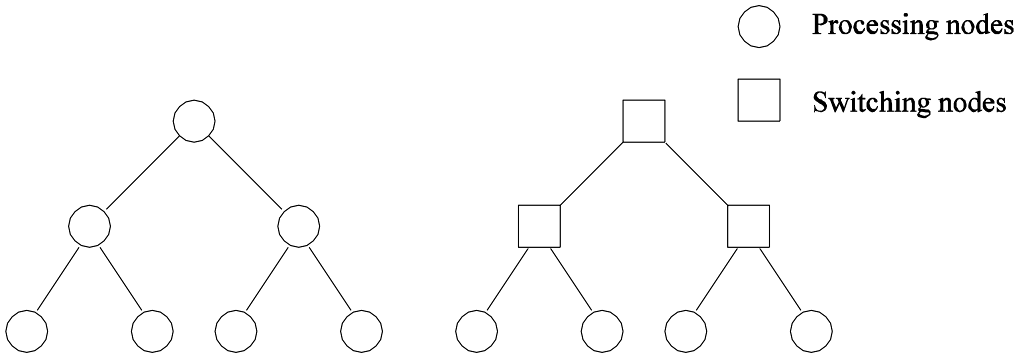


A 3-D mesh with no wraparound links

  - A variety of computations, e.g. physical simulations, can be mapped to 3-D topologies
    - 3-D weather modeling, structural modeling, etc.
    - Commonly used, e.g. Cray T3E, Jaguar/Titan machine at ORNL
- How is 3-D torus constructed?

Interconnect Topology: Tree-Based Networks

- Tree network is one in which only one path between any pair of nodes
  - Linear array network is a special case
- Static tree network: processing element at each node
- Dynamic tree network: nodes at intermediate level are switching nodes
- How is a message routed?
  - Route up to the root of the smallest subtree then routes down



(a) a static tree network

(b) a dynamic tree network

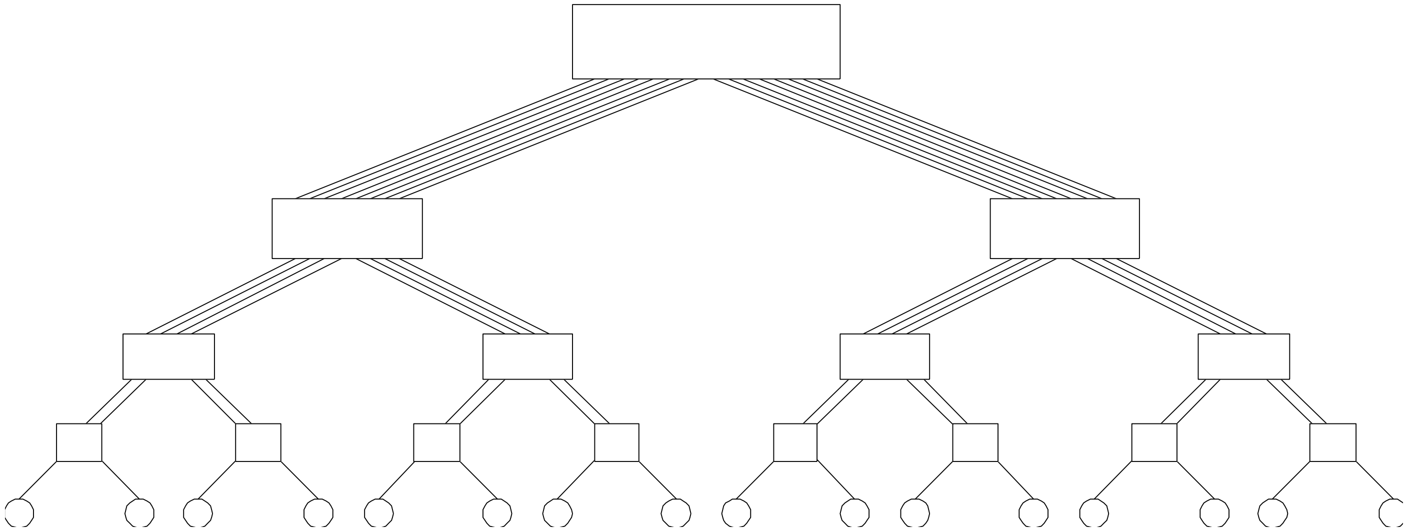
Interconnect Topology: Tree-Based Networks (cont.)

- The distance between any two nodes is no more than *2logp* (where *p* is the number of nodes)

- Links higher up the tree potentially carry more traffic than those at the lower levels.

- For this reason, a variant called a fat-tree, fattens the links as we go up the tree.

Interconnect Topology: Fat Trees



A fat tree network of 16 compute nodes

Evaluating Interconnect Topologies: Diameter

- Diameter: the maximum distance between any two nodes
  - The distance between two nodes is defined as the shortest path (in terms of the number of links) between them

- The diameter of a linear array is?
  $p - 1$

- Mesh? Max distnace bw nodes
  $2(\sqrt{p} - 1)$

- Tree (complete binary tree)?
  $2\log((p + 1)/2)\ \ or\ 2\,(\log(p + 1) - 1)$

Evaluating Interconnect Topologies: Arc Connectivity

- Connectivity: a measure of the multiplicity of paths between any two nodes
  - A network with high connectivity is desirable, because it lowers contention for communication resources

    How many different paths b/w any two nodes

- Arc connectivity: one measure of connectivity is the minimum number of arcs that must be removed from the network to break it into two disconnected networks

Evaluating Interconnect Topologies: Arc Connectivity

- Linear array?
  1
- Ring?
  2
- Mesh?
  2
- 2-D torus?
  4
- Tree?
  1

Evaluating Interconnect Topologies: Bisection Width

- Bisection Width: minimum number of links you must cut to divide the network into two equal parts

- The bisection width of a linear array and ring, respectively?

  1, 2

- Mesh, 2D-torus?
- $\sqrt{p}$
- $2\sqrt{p}$

- Tree?

  1

Evaluating Interconnect Topologies: Cost

- Cost: many criteria can be used to evaluate the cost of a network
- One way of defining the cost of a network is in terms of the number of communication links or the number of wires required by the network
- However, a number of other factors, such as the ability to layout the network, the length of wires, etc., also factor into the cost

Evaluating Interconnect Topologies: Cost (Number of links)

- Linear array, ring?

  $p - 1, p$

- 2-D mesh, 2-D torus

  $2(p - \sqrt{p})$

  $2p$

- Tree (complete binary tree)?

  $p - 1$

Readings

- Chapter 5, 5.1-5.4
- Chapter 6, 6.1

- POSIX Threads Programming, by Blaise Barney, Lawrence Livermore National Laboratory: https://hpc-tutorials.llnl.gov/posix/

- OpenMP Programming, by Blaise Barney, Lawrence Livermore National Laboratory: https://hpc-tutorials.llnl.gov/openmp/