Parallel Architecture

Sathish Vadhiyar

Motivations of Parallel Computing

- Faster execution times
  - From days or months to hours or seconds
  - E.g., climate modelling, bioinformatics
- Large amount of data dictate parallelism
- Parallelism more natural for certain kinds of problems, e.g., climate modelling
- Due to computer architecture trends
  - CPU speeds have saturated
  - Slow memory bandwidths

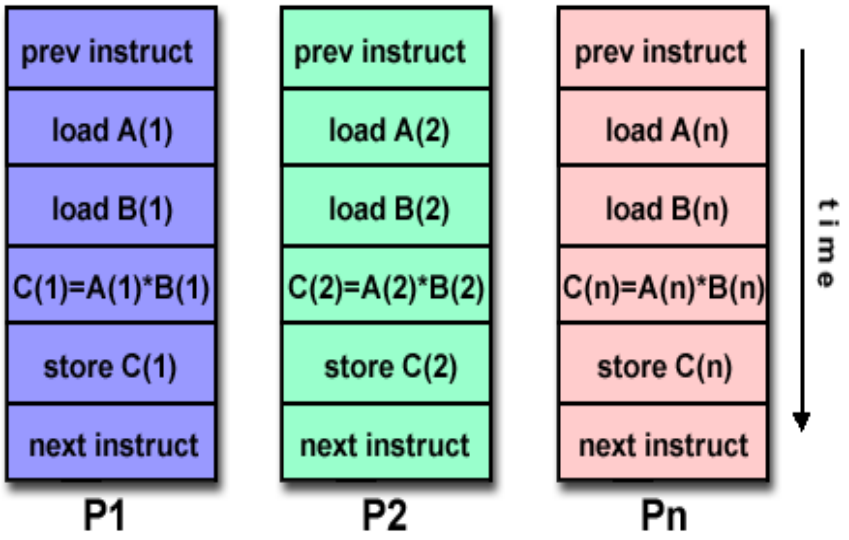
Classification of Architectures – Flynn's classification

In terms of parallelism in instruction and data stream

- Single Instruction Single Data (SISD): Serial Computers
- Single Instruction Multiple Data (SIMD)

  - Vector processors and processor arrays

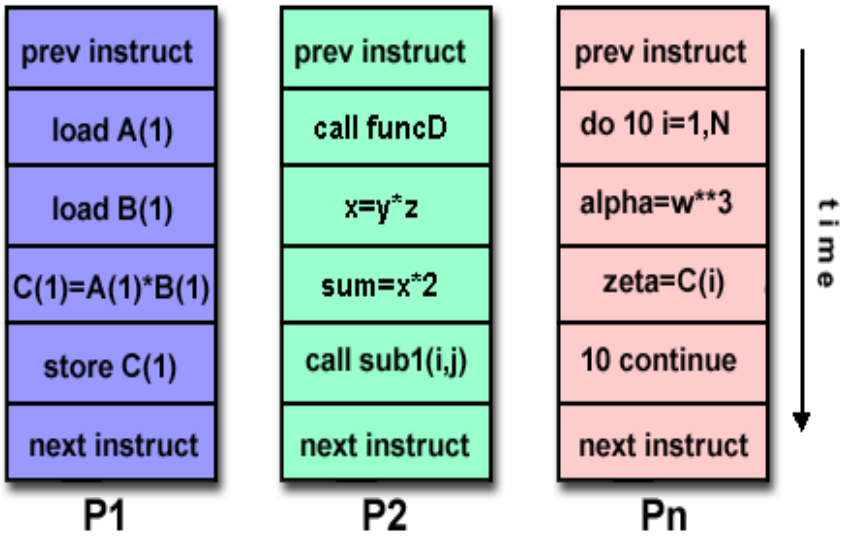
  - Examples: CM-2, Cray-90, Cray YMP, Hitachi 3600



Courtesy: http://www.llnl.gov/computing/tutorials/parallel_comp/

Classification of Architectures – Flynn's classification

- Multiple Instruction Single Data (MISD): Not popular
- Multiple Instruction Multiple Data (MIMD)
  - Most popular
  - IBM SP and most other supercomputers, clusters, computational Grids etc.

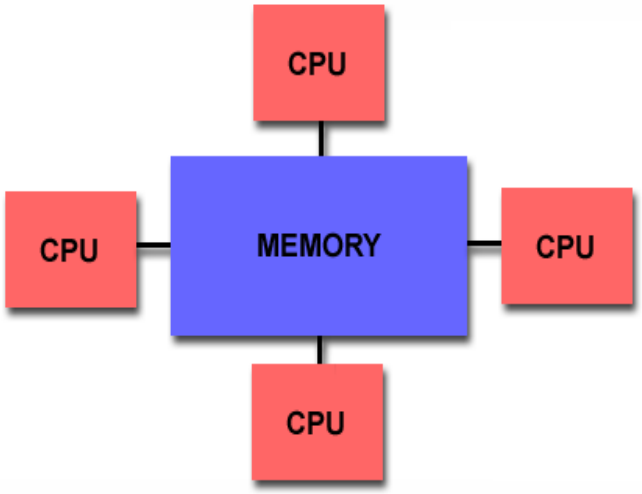


Courtesy: http://www.llnl.gov/computing/tutorials/parallel_comp/

Classification of Architectures – Based on Memory

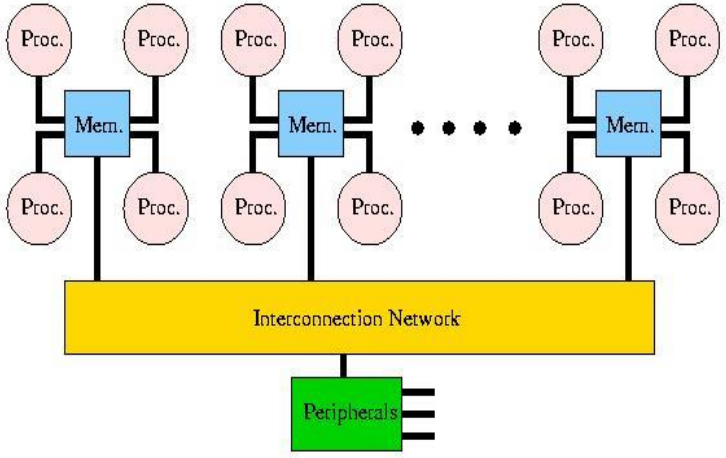
- Shared memory
- 2 types – UMA and NUMA

UMA



NUMA

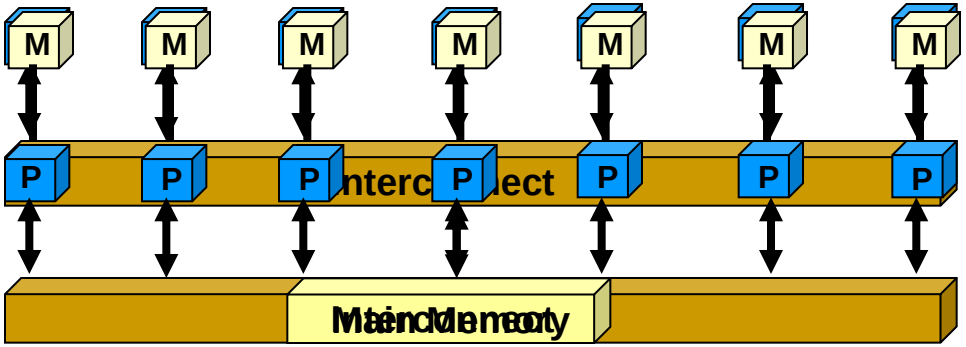
Examples: HP-Exemplar, SGI Origin, Sequent NUMA-Q



Courtesy: http://www.llnl.gov/computing/tutorials/parallel_comp/

Classification 2: Shared Memory vs Message Passing

- Shared memory machine: The n processors share physical address space
  - Communication can be done through this shared memory



- The alternative is sometimes referred to as a message passing machine or a distributed memory machine

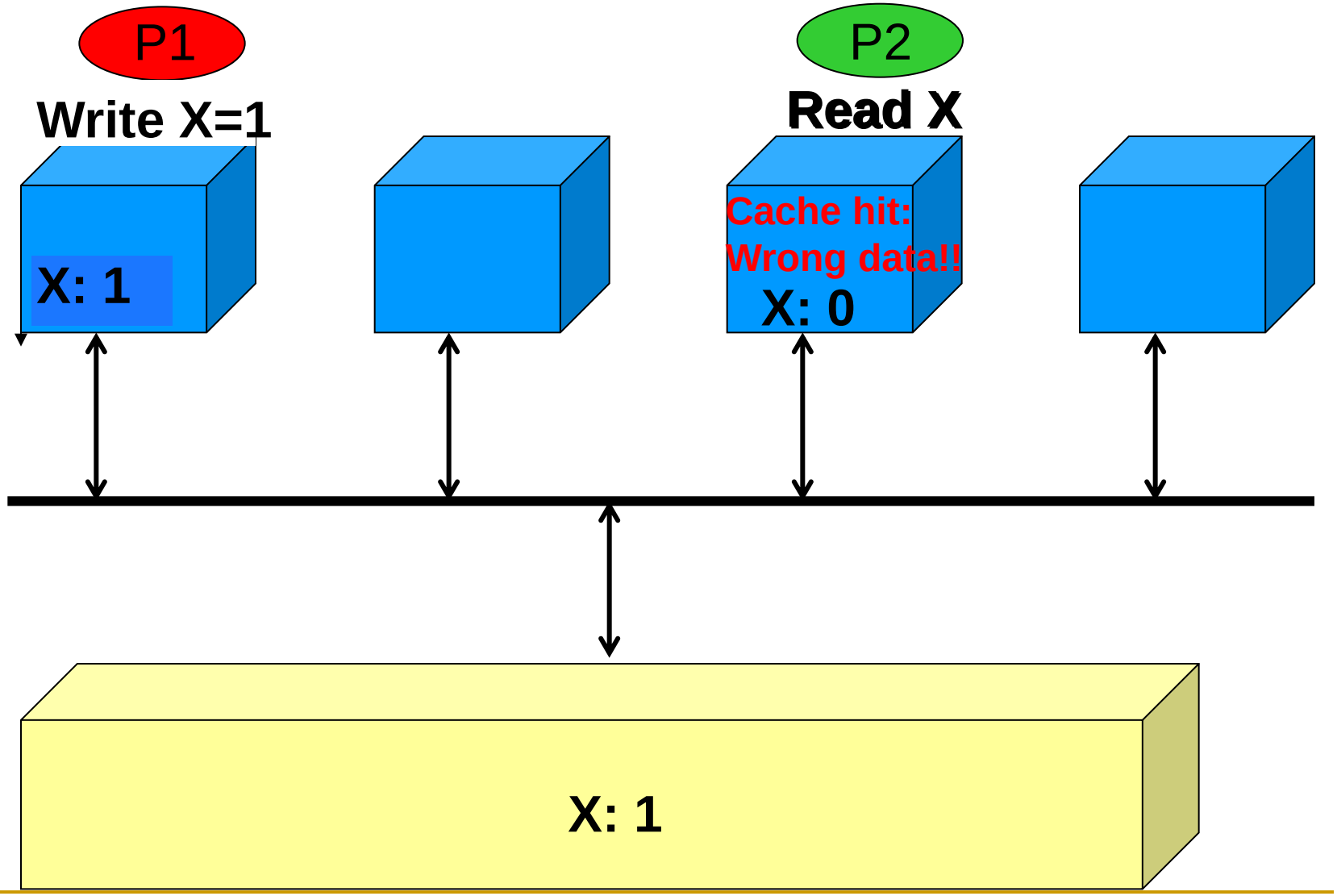
Shared Memory Machines

The shared memory could itself be distributed among the processor nodes

- Each processor might have some portion of the shared physical address space that is physically close to it and therefore accessible in less time
- Terms: NUMA vs UMA architecture
  - Non-Uniform Memory Access
  - Uniform Memory Access

SHARED MEMORY AND CACHES

Shared Memory Architecture: Caches



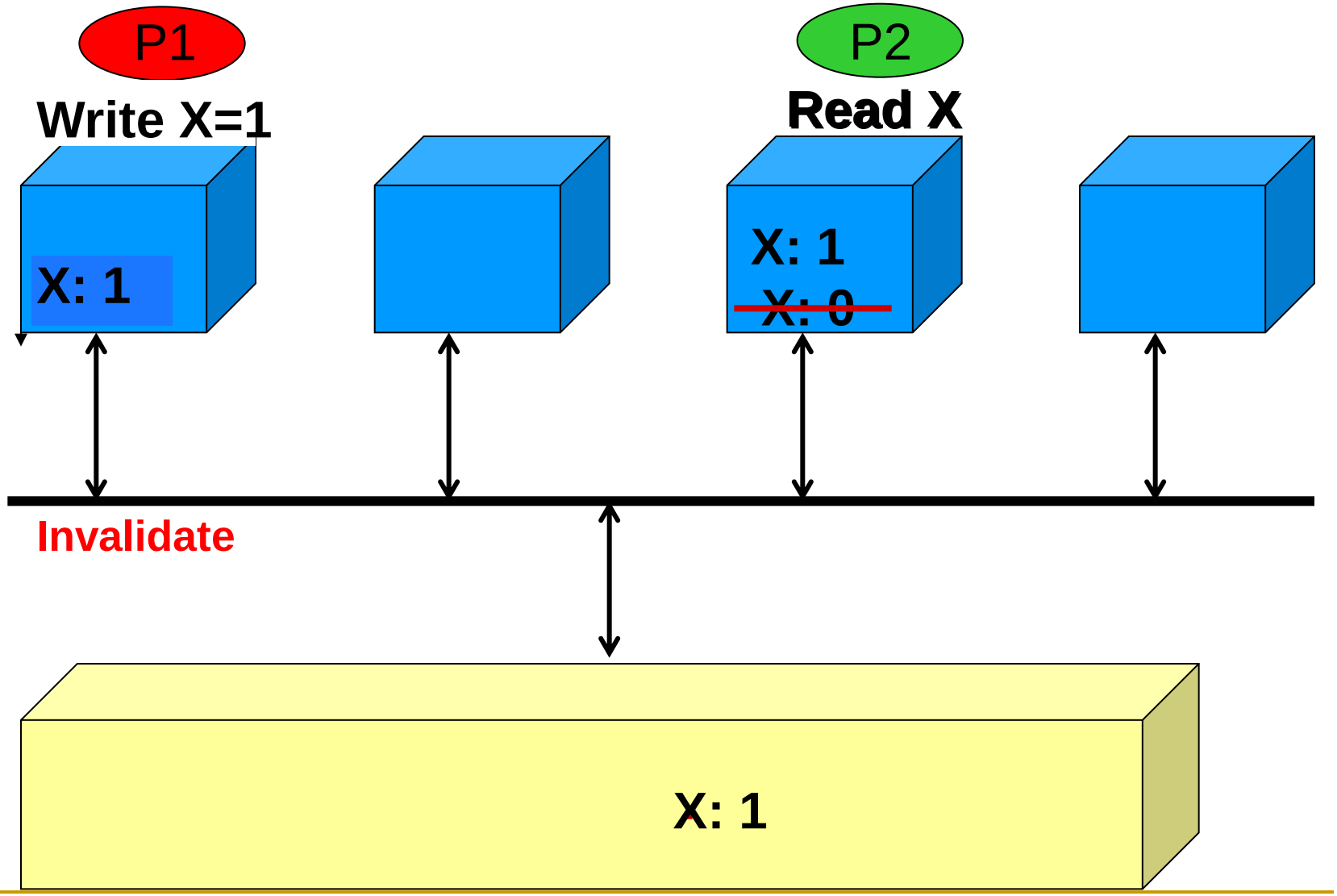
Cache Coherence Problem

- If each processor in a shared memory multiple processor machine has a data cache
  - Potential data consistency problem: the cache coherence problem
  - Shared variable modification, private cache
- Objective: processes shouldn't read `stale' data
- Solutions
  - Hardware: cache coherence mechanisms

Cache Coherence Protocols

- Write update – propagate cache line to other processors on every write to a processor
- Write invalidate – each processor gets the updated cache line whenever it reads stale data
- Which is better?

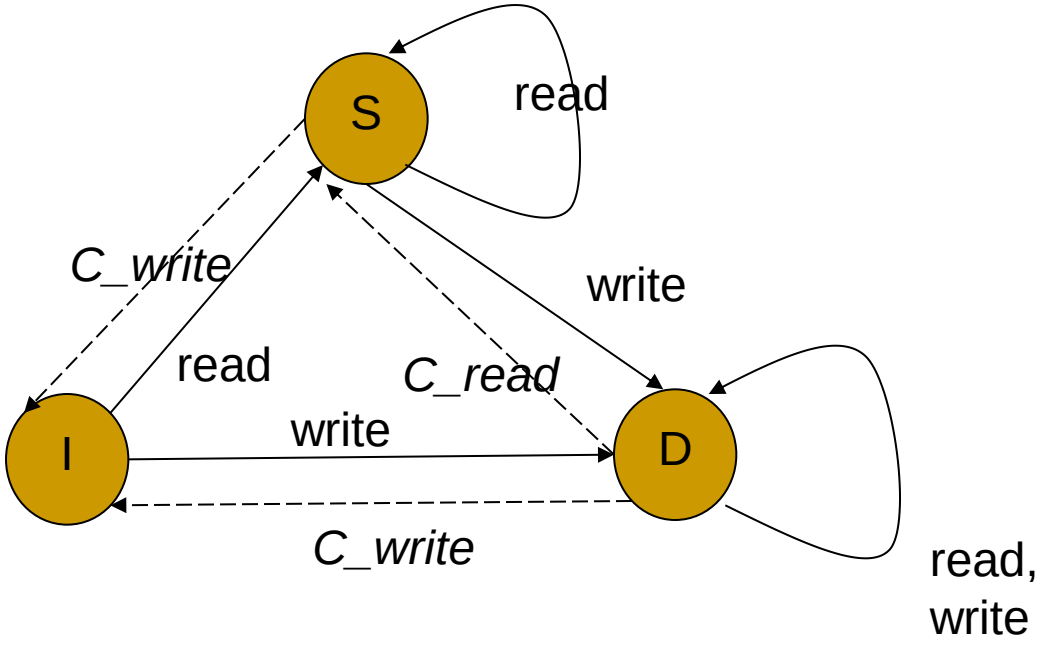
Invalidation Based Cache Coherence



Cache Coherence using invalidate protocols

- 3 states associated with data items
  - Shared – a variable shared by 2 caches
  - Invalid – another processor (say P0) has updated the data item
  - Dirty – state of the data item in P0

State Diagram



Implementations of cache coherence protocols

- Snoopy
  - for bus based architectures
  - shared bus interconnect where all cache controllers monitor all bus activity
  - There is only one operation through bus at a time; cache controllers can be built to take corrective action and enforce coherence in caches
  - Memory operations are propagated over the bus and snooped

Implementations of cache coherence protocols

- Directory-based
  - Instead of broadcasting memory operations to all processors, propagate coherence operations to relevant processors
  - A central directory maintains states of cache blocks, associated processors

Implementation of Directory Based Protocols

- Using presence bits for the owner processors
- Two schemes:
- *Full bit vector scheme* – O(MxP) storage for P processors and M cache lines
- But not necessary
- Modern day processors use *sparse or tagged directory scheme*
- Limited cache lines and limited presence bits

False Sharing

- Cache coherence occurs at the granularity of cache lines – an entire cache line is invalidated
- Modern day cache lines are 64 bytes in size
- Consider a Fortran program dealing with a matrix
- Assume each thread or process accessing a row of a matrix
- Leads to false sharing

False sharing: Solutions

- Reorganize the code so that each processor access a set of rows
- Can still lead to overlapping of cache lines if matrix size not divisible by processors
- In such cases, employ *padding*
- Padding: dummy elements added to make the matrix size divisible

INTERCONNECTION NETWORKS

Interconnects

- Used in both shared memory and distributed memory architectures
- In shared memory: Used to connect processors to memory
- In distributed memory: Used to connect different processors
- Components
  - *Interface* (PCI or PCI-e): for connecting processor to network link
  - *Network link* connected to a communication network (network of connections)

Communication network

- Consists of *switching elements* to which processors are connected through *ports*
- *Switch*: network of switching elements
- Switching elements connected with each other using a pattern of connections
- Pattern defines the *network topology*

- In shared memory systems, memory units are also connected to communication network

Parallel Architecture: Interconnections

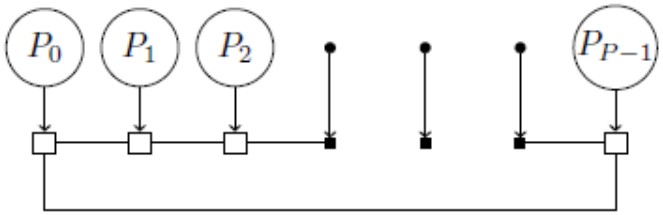
- Routing techniques: how the route taken by the message from source to destination is decided
- Network topologies
  - Static – point-to-point communication links among processing nodes
  - Dynamic – Communication links are formed dynamically by switches

Network Topologies

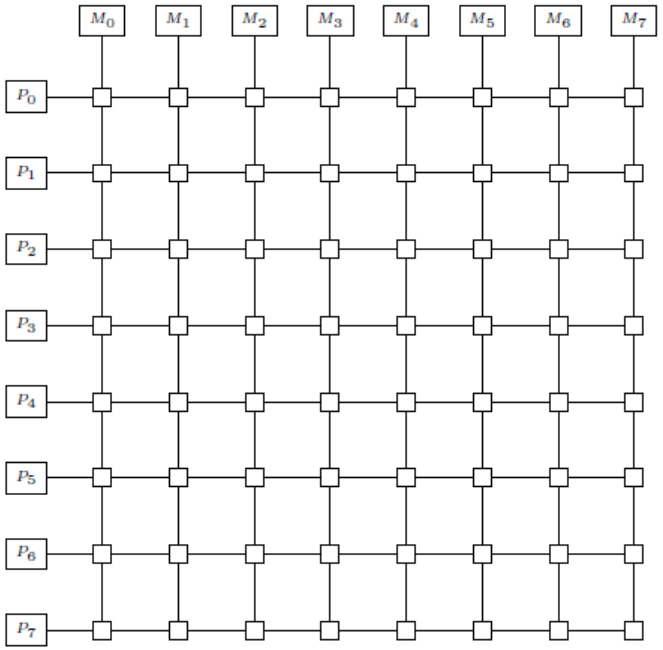
- Static
  - Bus
  - Completely connected
  - Star
  - Linear array, Ring (1-D torus)
  - Mesh
  - k-d mesh: d dimensions with k nodes in each dimension
  - Hypercubes – 2-logp mesh
  - Trees – our campus network
- Dynamic – Communication links are formed dynamically by switches
  - Crossbar
  - Multistage
- For more details, and evaluation of topologies, refer to book by Grama et al.

Network Topologies

- Bus, ring – used in small-scale shared memory systems

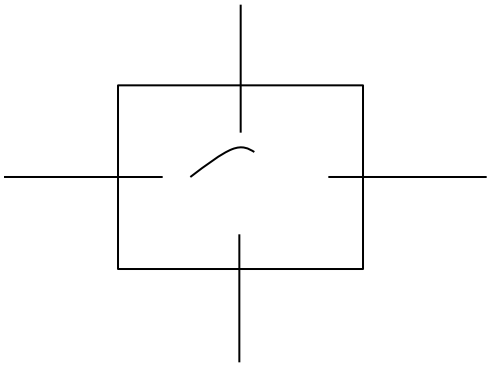


- Crossbar switch – used in some small-scale shared memory machines, small or medium-scale distributed memory machines



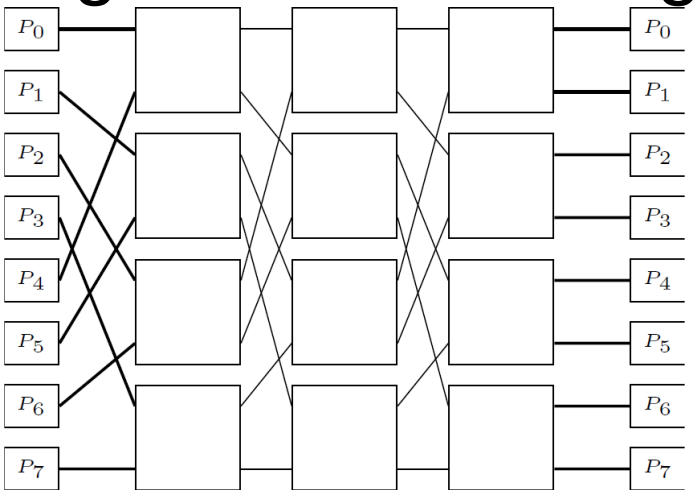
Crossbar Switch

- Consists of 2D grid of switching elements
- Each switching element consists of 2 input ports and 2 output ports
- An input port dynamically connected to an output port through a switching logic



Multistage network – Omega network

- To reduce switching complexity
- Omega network – consisting of logP stages, each consisting of P/2 switching elements

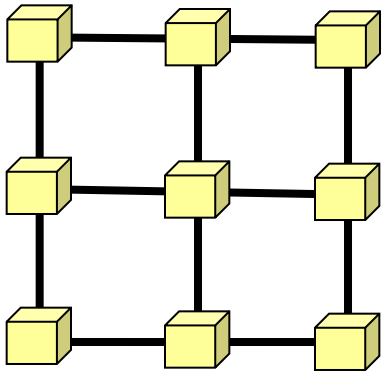


- Contention
  - In crossbar – nonblocking
  - In Omega – can occur during multiple communications to disjoint pairs

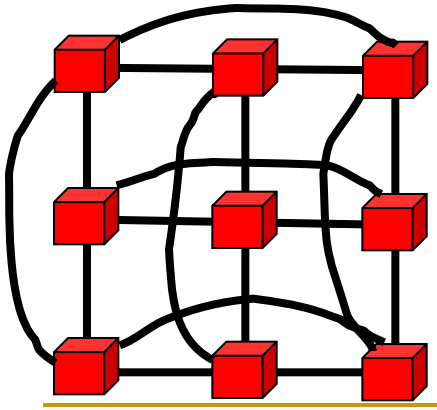
Mesh, Torus, Hypercubes, Fat-tree

- Commonly used network topologies in distributed memory architectures
- Hypercubes are networks with dimensions

Mesh, Torus, Hypercubes



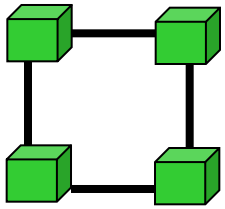
2D
Mesh



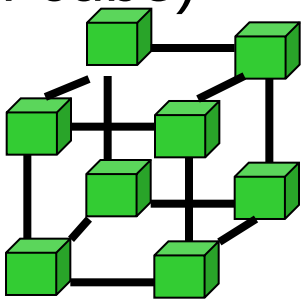
Torus

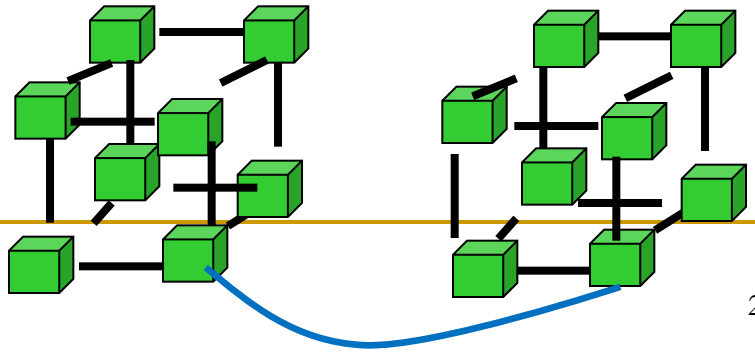
Hypercube (binary n-cube)

n=2



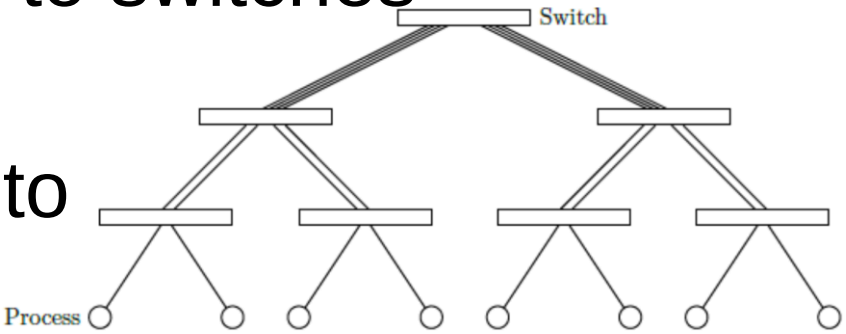
n=3





Fat Tree Networks

- Binary tree
- Processors arranged in leaves
- Other nodes correspond to switches
- Fundamental property:
  No. of links from a node to a children = no. of links from the node to its parent
- Edges become fatter as we traverse up the tree



Fat Tree Networks

- Any pairs of processors can communicate without contention: non-blocking network
- Constant Bisection Bandwidth (CBB) networks
- Two level fat tree has a diameter of four

Evaluating Interconnection topologies

- Diameter – maximum distance between any two processing nodes
  - Full-connected – 1
  - Star – 2
  - Ring – p/2
  - Hypercube - logP
- Connectivity – multiplicity of paths between 2 nodes. Miniimum number of arcs to be removed from network to break it into two disconnected networks
  - Linear-array – 1
  - Ring – 2
  - 2-d mesh – 2
  - 2-d mesh with wraparound – 4
  - D-dimension hypercubes – d

Evaluating Interconnection topologies

- bisection width – minimum number of links to be removed from network to partition it into 2 equal halves
  - Ring – 2
  - P-node 2-D mesh - Root(P)
  - Tree – 1
  - Star – 1
  - Completely connected – $P^2/4$
  - Hypercubes - P/2

Evaluating Interconnection topologies

- channel width – number of bits that can be simultaneously communicated over a link, i.e. number of physical wires between 2 nodes
- channel rate – performance of a single physical wire
- channel bandwidth – channel rate times channel width
- bisection bandwidth – maximum volume of communication between two halves of network, i.e. bisection width times channel bandwidth