




PARALLEL PROCESSING SYSTEMS

Chapter 4: Models of Parallel Processing



Introduction

- we deal with abstract models of real machines
 - Benefits
 - technology-independent theories
 - algorithmic techniques that are applicable to many existing and future machines
 - Disadvantages
 - inability to predict the actual performance accurately
 - tendency to simplify the models too much, so that they no longer represent any real machine.
- 



Development of early models

- Associative processing (AP)
 - perhaps the earliest form of parallel processing
 - Associative or content-addressable memories (AMs, CAMs)
 - allow memory cells to be accessed based on contents
 - came into the forefront in the 1950s when advances in technologies
 - the origins of research go back to the 1943



Development of early models

- Associative processing (AP)
 - Early memories
 - provided two basic capabilities
 - masked search
 - looking for a particular bit pattern in selected fields of all memory words
 - marking those for which a match is indicated
 - parallel write
 - storing a given bit pattern into selected fields of all memory words that have been previously marked.
 - suffice for
 - programming of sophisticated searches
 - even parallel arithmetic operations






Development of early models

- Perceptrons
 - introduced in the 1950s
 - a neuronlike device in charge of processing a single pixel in a digital image
 - laid the foundation for the modern field of neural networks




Development of early models

- Cellular automata
 - a model of fundamental importance introduced in 1960
 - natural extensions von Neumann-type sequential computers
 - typically viewed as a collection of identical finite-state automata
 - are interconnected, through their input–output links, in a regular fashion
 - state transitions controlled by
 - its own state
 - the states of the connected neighbors
 - its primary inputs, if any [Garz95].
 - Systolic arrays can be viewed as cellular automata
 - basis of high-performance VLSI-based designs
 - In recent years interest in cellular automata
 - as theoretical models of massively parallel systems
 - as tools for modeling physical phenomena
- 



SIMD versus MIMD architectures

- Most early parallel machines had SIMD designs
 - a central unit
 - fetches and interprets the instructions
 - broadcasts appropriate control signals to several processors operating in lockstep.
- 



SIMD versus MIMD architectures

- Two fundamental design choices in SIMD:
 - Synchronous versus asynchronous
 - Custom- versus commodity-chip



SIMD versus MIMD architectures

- Two fundamental design choices in SIMD:
 - Synchronous versus asynchronous
 - each processor can execute or ignore the instruction
 - based on its local state or data-dependent conditions
 - leads to inefficiency in conditional computations
 - “if-then-else” statement
 - is executed by
 - first enabling the processors for which the condition is satisfied
 - then flipping the “enable” bit before getting into the “else” part.
 - On the average, half of the processors will be idle for each branch.
 - situation is even worse for “case” statements



SIMD versus MIMD architectures

- Two fundamental design choices in SIMD:
 - Synchronous versus asynchronous
 - A possible cure is to use the asynchronous version known as SPMD
 - where each processor runs its own copy of the common program
 - The advantage in an “if-then-else” computation
 - each processor will only spend time on the relevant branch
 - The disadvantages include the need for
 - occasional synchronization
 - higher complexity of each processor
 - must have a program memory and instruction fetch/decode logic.




SIMD versus MIMD architectures

- Two fundamental design choices in SIMD:
 - Custom versus commodity-chip
 - designed can be based on
 - commodity (off-the-shelf) components
 - Components are general purpose
 - tend to be inexpensive because of mass production
 - will likely contain unwanted elements
 - may complicate the design, manufacture, and testing of the SIMD machine
 - may introduce speed penalties as well
 - custom chips
 - generally, offer better performance
 - but lead to much higher cost




SIMD versus MIMD architectures

- MIMD has become more popular recently
 - Because of
 - higher flexibility
 - their ability to take advantage of commodity microprocessors
 - avoiding lengthy development cycles
 - getting a free ride on the speed improvement curve for such microprocessors
 - MIMD are most effective for medium to coarse-grain parallel applications
 - the computation is divided into relatively large subcomputations or tasks
 - executions are assigned to the various processors.
- 




SIMD versus MIMD architectures

- Three fundamental design choices in MIMD:
 - MPP—massively or moderately parallel processor
 - Tightly versus loosely coupled MIMD
 - Explicit message passing versus virtual shared memory
- 



SIMD versus MIMD architectures

- Three fundamental design choices in MIMD:
 - MPP—massively or moderately parallel processor
 - the “herd of elephants” or the “army of ants” approach?
 - A general answer cannot be given to this question
 - the best choice is both application- and technology-dependent
- 



SIMD versus MIMD architectures

- Three fundamental design choices in MIMD:
 - Tightly versus loosely coupled MIMD
 - Which is a better approach
 - Using specially designed multiprocessors/ multicomputers
 - using network of workstations (cluster computing)
 - collection of ordinary workstations
 - interconnected by commodity networks (such as Ethernet or ATM)
 - whose interactions are coordinated by special system software and distributed file systems
 - has been gaining popularity in recent years
 - However, many open problems exist for taking full advantage of such architectures



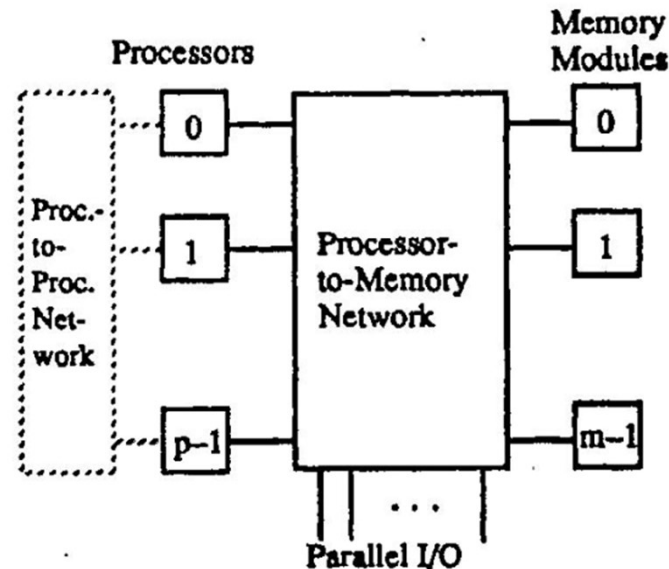
SIMD versus MIMD architectures

- Three fundamental design choices in MIMD:
 - Explicit message passing versus virtual shared memory
 - Which scheme is better
 - forcing the users to explicitly specify all messages that must be sent between processors
 - allow users to program in an abstract higher-level model
 - The required messages automatically generated by the system software



Global versus distributed memory

- Global memory
 - a central location where all processors can access it with equal ease
 - a special processor-to-memory network
 - must have very low latency
 - optional processor-to-processor network
 - for coordination and synchronization purposes





Global versus distributed memory

- Global memory multiprocessor
 - Is characterized by
 - the type and number p of processors
 - the capacity and number m of memory modules
 - and the network architecture
 - p and m are independent parameters
 - But must be comparable in magnitude
 - too few memory modules causes contention among the processors
 - too many would complicate the network design.

Global versus distributed memory

- Global memory multiprocessor

- Example networks

1. Crossbar switch

1. $O(pm)$ complexity

2. quite costly for highly parallel systems

2. Single or multiple buses

3. Multistage interconnection network (MIN)

1. cheaper than Example 1

2. more bandwidth than Example 2

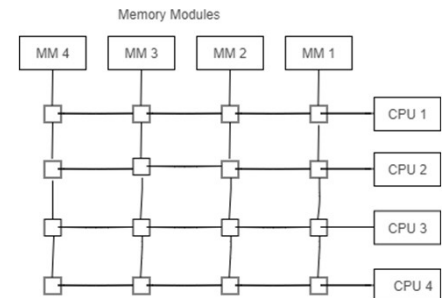
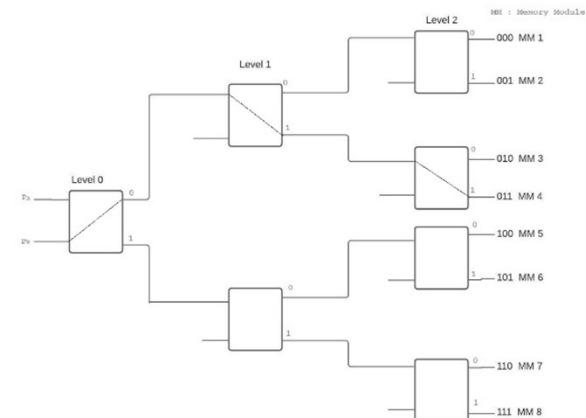


Figure - Crossbar Switch System



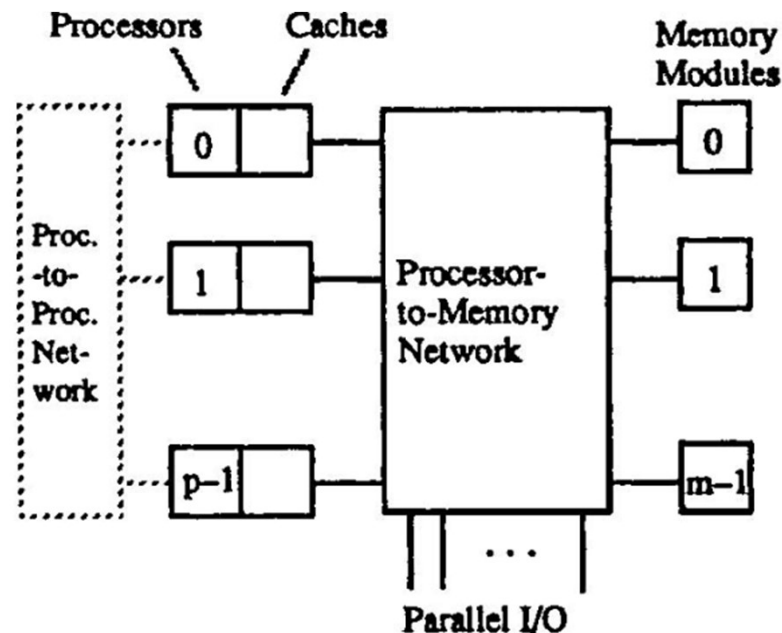


Global versus distributed memory

- Global memory multiprocessor
 - network type affects efficient algorithm development
 - programmers should be aware of tedious considerations
 - PRAM abstract model to reduce difficulties

Global versus distributed memory

- Global memory multiprocessor
 - private cache memory
 - reasonable size memory within each processor
 - is used to reduce the processor-to-memory traffic
 - Because of locality of data access



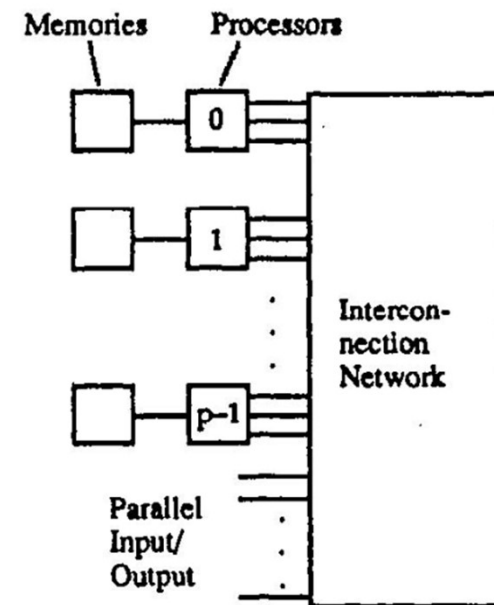


Global versus distributed memory

- Global memory multiprocessor
 - private cache memory
 - cache coherence problem in the case of multiple caches
 - Different tackling approaches
 1. Do not cache shared data at all
 - Works for small volume shared data with infrequent access to it
 2. Do not cache “writeable” shared data
 3. Use a cache coherence protocol
 - may introduce a nontrivial overhead
 - Examples
 - snoopy caches for bus-based systems
 - each cache monitors all data transfers on the bus
 - directory-based schemes
 - writeable shared data are “owned” by a single processor or cache at any given time
 - a directory is used to determine physical data locations

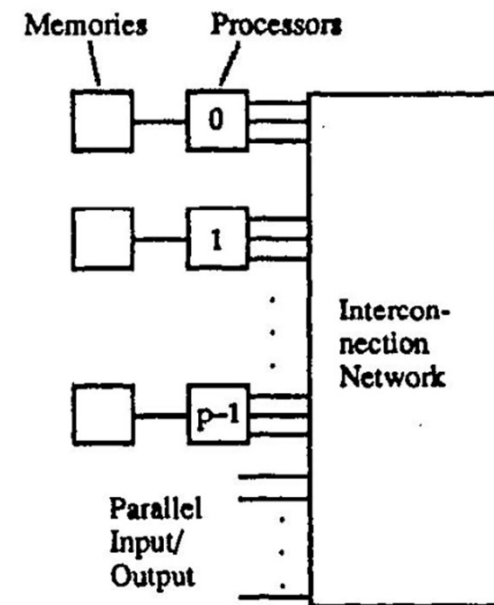
Global versus distributed memory

- Distributed-memory architectures
 - A collection of p processors
 - each with its own private memory
 - communicates through a network
 - the latency of the network may be less critical
 - each processor is likely to access its own local memory most of the time
 - the bandwidth of the network may or may not be critical depending on
 - the type of parallel applications
 - the extent of task interdependencies




Global versus distributed memory

- Distributed-memory architectures
 - each processor usually has multiple links to network
 - Also known as nonuniform memory access (NUMA) architectures
 - access to remote memory modules involves considerably more latency
 - inattention to data and task partitioning may have dire consequences
 - load-balancing is also of some importance





The PRAM shared-memory model

- RAM (random-access machine)
 - The theoretical model used for SISD class
 - not to be confused with random-access memory
 - PRAM
 - The parallel version of RAM
 - an abstract model of the class of global-memory parallel processors
- 

The PRAM shared-memory model

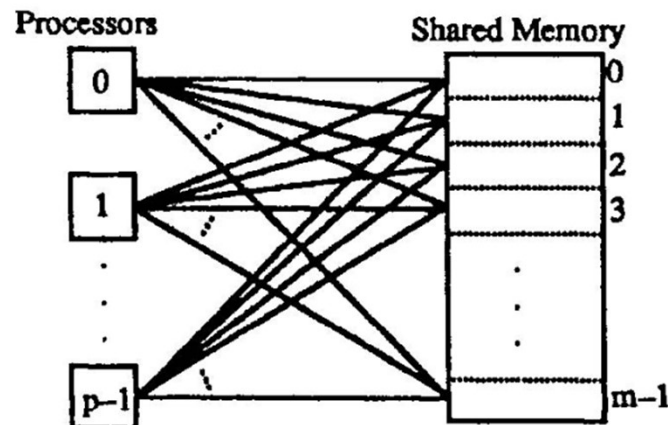
- In the formal PRAM model
 - a single processor is assumed to be active initially
 - In each computation step, each active processor
 - can read from and write into the shared memory
 - can also activate another processor.
 - $\lceil \log_2 p \rceil$ steps are necessary to activate all p processors.
- In our discussions
 - the set of active processors is usually implied
 - We do not explicitly activate the processors.

The PRAM shared-memory model

- PRAM algorithms might involve statements like
 - “for $0 \leq i < p$, Processor i adds the contents of memory location $2i + 1$ to the memory location $2i$ ”
 - different locations accessed by the various processors
 - “each processor loads the contents of memory location x into its Register 2”
 - the same location accessed by all processors
- The problem of multiple processors attempting to write into a common memory location must be resolved in some way.
 - A detailed discussion of this issue is postponed to Chapter 5

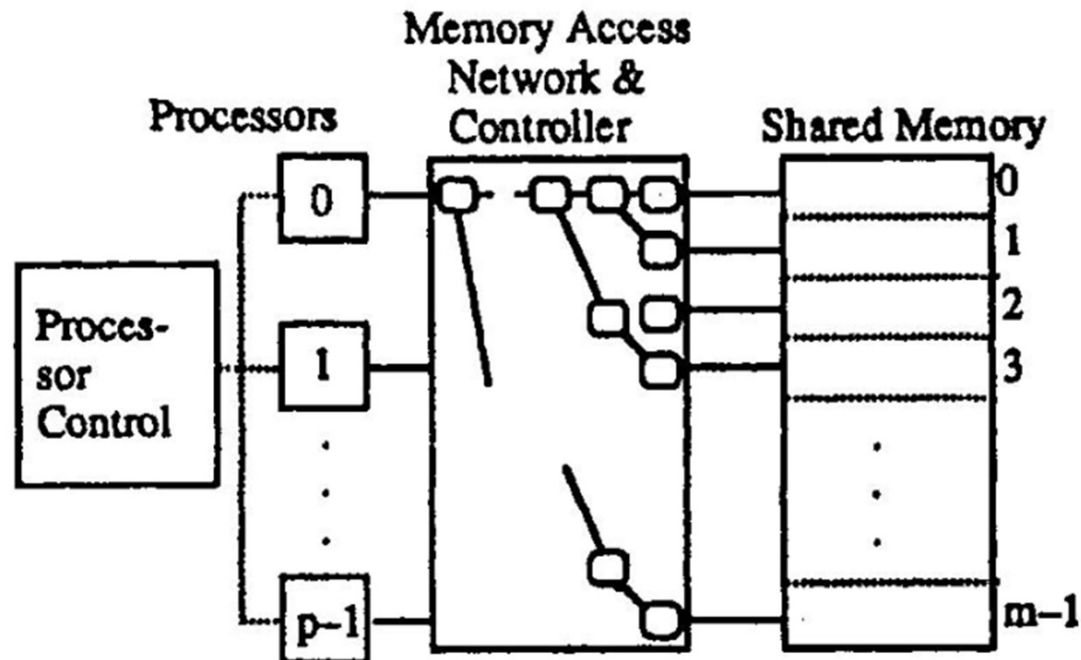
The PRAM shared-memory model

- can also be SIMD
 - all processors obey the same instruction in each machine cycle
 - they often execute the operation on different data
 - because of indexed and indirect (register-based) addressing



The PRAM shared-memory model

- the previous depicted model is highly theoretical.
 - processor-to-memory connectivity would have to be realized by an interconnection network.





Distributed-memory or graph models

- is characterized primarily by its network
 - usually represented as a graph
 - vertices corresponding to processor–memory nodes
 - edges corresponding to communication links.
 - directed edges if communication links are unidirectional
 - undirected edges for bidirectional communication





Distributed-memory or graph models

- is characterized primarily by its network
 - Important parameters include
 - Network diameter
 - the longest of the shortest paths between various pairs of nodes
 - should be relatively small if network latency is to be minimized
 - is more important with store-and-forward routing
 - a message is stored in its entirety and retransmitted by intermediate nodes
 - Is less important with wormhole routing
 - a message is quickly relayed through a node in small pieces
 - Bisection (band)width
 - Vertex or node degree



Distributed-memory or graph models

- is characterized primarily by its network
 - Important parameters include
 - Network diameter
 - Bisection (band)width:
 - the smallest number (total capacity) of links that need to be cut in order to divide the network into two subnetworks of half the size.
 - is important when nodes communicate with each other in a random fashion.
 - A small bisection (band)width
 - limits the rate of data transfer between the two halves of the network
 - affects the performance of communication-intensive algorithms.
 - Vertex or node degree



Distributed-memory or graph models

- is characterized primarily by its network
 - Important parameters include
 - Network diameter
 - Bisection (band)width
 - Vertex or node degree:
 - the number of communication ports required for each node
 - should be a constant independent of network size
 - For scalability of the architecture
 - has a direct effect on the cost of each node
 - more significant when the node is required to communicate over all of its ports at once.