Introduction to Multiprocessors (Part I)

Prof. Cristina Silvano
Politecnico di Milano
Outline

- Key issues to design multiprocessors
- Interconnection network
- Centralized shared-memory architectures
- Distributed shared-memory architectures
- Message passing architectures
- Communication models
- Introduction to the problem of cache coherence
Multiprocessors

- **A new era in Computer Architecture**
  - Multiprocessors now play a major role from embedded to high end general-purpose computing:
    - Main goals: very high-end performance; scalability; reliability.
  - Multiprocessors refers to computers consisting usually of tightly coupled processors whose coordination and usage is controlled by a single operating system and that usually share memory through a shared address space
  - Multicores/Manycores when all cores are in the same chip
Key issues to design multiprocessors

- How many processors?
- How powerful are processors?
- How do connect processors?
- How do parallel processors share data?
- Where to place the physical memory?
- How do parallel processors cooperate and coordinate?
- How to program processors?
- How to maintain cache coherence?
- How to maintain memory consistency?
- How to evaluate system performance?
How do connect processors?

Single bus vs. interconnection network:

- Single-bus approach imposes constraints on the number of processors connected to it (up to now, 36 is the largest number of processors connected in a commercial single bus system) ⇒ saturation.

- To connect many processors with a high bandwidth, the system needs to use more than a single bus ⇒ introduction of an interconnection network.
In Single-Bus Multiprocessors, the connection medium (*the bus*) is between the processors and memory ⇒ the medium is used on every memory access.
In **Network-Connected Multiprocessors**, memory is attached to each processor, and the connection medium *(the network)* is between the nodes ⇒ the medium is used only for interprocessor communication.
Network Topologies

- Between the high cost/performance of a **fully connected network** (with a dedicated communication link between every node) and the low cost/performance of a **single bus**, there are a set of network topologies that constitutes a wide range of trade-offs in cost/performance metric.

- Examples of topology of the interconnection network:
  - Single bus
  - Ring
  - Mesh
  - N-cube
  - Crossbar Network

- The topology of the interconnection network can be different for **data** and **address** buses.
Network Representation and Costs

- Networks are represented as **graphs** where:
  - **Nodes** (shown as black square) are processor-memory nodes
  - **Switch** (shown as red circle) whose links go to processor-memory nodes and to other switches
  - **Arcs** representing a link of the communication network (all links are bidirectional $\Rightarrow$ information can flow in either direction).

- **Network costs** include:
  - Number of switches
  - Number of links on a switch to connect to the network
  - Width (number of bits) per link
  - Length of links when the network is mapped to a physical machine
Single-bus and Ring

- Single-bus

- Ring: capable of many simultaneous transfers (like a segmented bus). Some nodes are not directly connected ⇒ the communication between some nodes needs to pass through intermediate nodes to reach the final destination (multiple-hops).
Network Performance Metrics

- **Total Network Bandwidth** (*best case*): bandwidth of each link multiplied by the number of links.
  
  \[ P = \text{number of nodes} \]
  
  \[ b = \text{bandwidth of a single link} \]

- For a single-bus topology, the total network bandwidth is just the bandwidth of the bus, or 1 times the bandwidth of the link \((1 \times b)\)

- For a ring topology, the total network bandwidth is \(P\) times the bandwidth of one link \((P \times b)\)
Network Performance Metrics

- **Bisection Bandwidth** *(worst case)*: This is calculated by dividing the machine into two parts, each with half the nodes. Then you sum the bandwidth of the links that cross that imaginary dividing line.

- The bisection bandwidth of a ring is 2 times the link bandwidth \((2 \times b)\), while for the single bus it is 1 times the link bandwidth \((1 \times b)\).

- Some network topologies are not symmetric: where to draw the bisection line? To find the worst case metric, we need to choose the division that yields the most pessimistic network performance (that is calculate all possible bisection bandwidth and pick the smallest).
Bisection Bandwidth Calculation

- **Single-bus**

- **Ring**
### Single-bus vs Ring

<table>
<thead>
<tr>
<th></th>
<th>Best case</th>
<th>Worst case</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Bandwidth</td>
<td>Bisection Bandwidth</td>
</tr>
<tr>
<td>Single Bus</td>
<td>$b$</td>
<td>$b$</td>
</tr>
<tr>
<td>Ring</td>
<td>$P \cdot b$</td>
<td>$2b$</td>
</tr>
</tbody>
</table>

- Single-bus is as fast as a single link.
- Given the bandwidth $b$ of a single link:
  - The ring is only *twice* as fast as the single-bus in the worst case.
  - The ring is $P$ times faster in the best case.
Crossbar Network

- **Crossbar Network or fully connected network:** every processor has a bidirectional dedicated communication link to every other processor
  - very high cost

- **Total Bandwidth:** \( \left(\frac{P \times (P - 1)}{2}\right) \times b \)

- **Bisection Bandwidth:** \( \frac{P}{2} \times b \)

- **Example:** \( P = 4 \) nodes
  - Total BW = \( 6 \times b \); Bisection BW = \( 4 \times b \)
Bidimensional Mesh (2D Mesh)

- Given P nodes: \( N = \sqrt{P} \)
- \( N \times (N - 1) \) horizontal channels
- \( N \times (N - 1) \) vertical channels
- Number of links per internal switch = 5
- Number of links per external switch = 3
- Total Bandwidth: \( \{2N(N-1)\} \times b \)
- Bisection Bandwidth: \( N \times b \)
- Example: \( P = 4; N = 2 \Rightarrow \text{Total Bandwidth} = 4 \times b \), Bisection Bandwidth = \( 2 \times b \)
Bidimensional Mesh (2D Mesh)

Example: $P = 16; \ N = 4 \Rightarrow$
- 12 horizontal channels
- 12 vertical channels
Total Bandwidth = $24 \times b$; Bisection Bandwidth = $4 \times b$
Hypercube

- Boolean N-cube with $P = 2^N$ nodes.
- Each node has $N$ neighborhood nodes.
- Number of links per switch = $N+1$
- Total Bandwidth: $\{(N \times P) / 2 \} \times b$
- Bisection Bandwidth: $2^{(N-1)} \times b$
- Example: $P = 4$ $N = 2 \Rightarrow$ Total Bandwidth = $4 \times b$; Bisection Bandwidth = $2 \times b$
Hypercube

- Example: \( P = 8 \) \( N = 3 \) \( \Rightarrow \) Total Bandwidth = \( 12 \times b \),
  Bisection Bandwidth = \( 4 \times b \)
Hypercube

Example: \( P = 16 \ N = 4 \Rightarrow \) Total Bandwidth = \( 32 \times b \),
Bisection Bandwidth = \( 8 \times b \)
How do parallel processors share data?

- Memory Address Space Model:
  - Single logically shared address space
  - Multiple and private address spaces
Memory Address Space Model

- **Single logically shared address space:** A memory reference can be made by any processor to any memory location ⇒ *Shared Memory Architectures.*
  - The address space is shared among processors: The same physical address on 2 processors refers to the same location in memory.

- **Multiple and private address spaces:** The processors communicate among them through send/receive primitives ⇒ *Message Passing Architectures.*
  - The address space is logically disjoint and cannot be addressed by different processors: the same physical address on 2 processors refers to 2 different locations in 2 different memories.
Shared Addresses

- The processors communicate among them through shared variables in memory.
- In shared memory architectures, communication among threads occurs through a shared address space.
- Implicit management of the communication through load/store operations to access any memory locations.
  - Oldest and most popular model
- Shared memory does not mean that there is a single centralized memory. The shared memory can be centralized or distributed over the nodes.
- Shared memory model imposes the cache coherence problem among processors.
Multiple and Private Addresses

- The processors communicate among them through sending/receiving messages: message passing protocol
- Explicit management of the communication through send/receive primitives to access private memory locations.
- The memory of one processor cannot be accessed by another processor without the assistance of software protocols.
- No cache coherence problem among processors.
Where to place the physical memory?

- Physical Memory Organization:
  - Centralized Memory
  - Distributed Memory

Prof. Cristina Silvano, Politecnico di Milano
Physical Memory Organization

- **Centralized Memory**
  - **UMA (Uniform Memory Access):** The access time to a memory location is *uniform* for all the processors: no matter which processor requests it and no matter which word is asked.

- **Distributed Memory**
  - The physical memory is divided into memory modules distributed to the single processors.
  - **NUMA (Non Uniform Memory Access):** The access time to a memory location is *non uniform* for all the processors: it depends on the location of the data word in memory and the processor location.
Physical Memory Organization

- Multiprocessor systems can have single addressing space and distributed physical memory.
- The concepts of addressing space (single/multiple) and the physical memory organization (centralized/distributed) are orthogonal to each other.

Address Space

Physical Memory Organization
Address Space vs. Physical Mem. Org.

**Address Space**
- Single Logically Shared Address Space
  (Shared-Memory Architectures)

**Physical Memory Organization**
- **Centralized Memory**
- **Distributed Memory**

**UMA**
- N0
- P0
- CO
- MM0

**NUMA**
- N3
- P3
- C3
- MM3

Prof. Cristina Silvano, Politecnico di Milano
Address Space vs. Physical Mem. Org.

Physical Memory Organization

- **Centralized Memory**
- **Distributed Memory**

**Address Space**
- Multiple and Private Address Spaces (Message Passing Architectures)

**Centralized Memory**
- N0
- N3
- P0
- P3
- CO
- C3
- MM0
- MM1
- MM2
- MM3

**Distributed Memory**
- N0
- N3
- P0
- P3
- C0
- C3
- MM0
- MM3

**UMA**

**NUMA**

Prof. Cristina Silvano, Politecnico di Milano
Address Space vs. Physical Mem. Org.

- Centralized Memory
- Distributed Memory

- Single Logically Shared Address Space (Shared-Memory Architectures)
- Multiple and Private Address Spaces (Message Passing Architectures)
Centralized Shared-Memory (CSM) Arch.

- **UMA (Uniform Memory Access):** Memory access time is uniform for all processors and all memory locations.
- **CSM multiprocessors are also called as Symmetric Multiprocessors (SMPs)**

Prof. Cristina Silvano, Politecnico di Milano
Small-Scale MP Designs

- **SMP** architectures with **uniform memory access time (UMA)** and **single BUS**.
- Snooping protocols to support cache coherence

![Diagram of small-scale MP designs with nodes, processors, caches, single bus, main memory, and I/O system.](Diagram.png)
Most of existing **multicores** are typically **SMPs**

- **Small number of cores (<=8)**
  - Share a single centralized memory with uniform memory access (UMA)
  - There is typically one level of shared cache, and one or more levels of private per-core cache
Multicore single-chip multiprocessor

- As the number of processors grows (>8), any centralized resource in the system becomes a bottleneck.

- To increase the communication bandwidth -> multiple buses as well as interconnection networks (such as crossbars or small point-to-point networks), where memory or a shared cache can be configured into multiple physical banks.

- Basic idea: to boost the effective memory bandwidth while retaining uniform access time to memory.

Prof. Cristina Silvano, Politecnico di Milano
Multicore single-chip multiprocessor

- Multicore single-chip multiprocessor with UMA through a banked shared cache and using an interconnection network rather than a bus.
Distributed Shared-Mem. (DSM) Arch.

- Scalable Shared-Memory Architectures with a larger processor count (from 8 to 32)
- **NUMA (Non Uniform Memory Access):** Memory access time depends on processor and memory location.
- **CC-NUMA (Cache Coherent NUMA)**

- Each processor core shares the entire memory space
- Access time to the memory attached to the processor’s node will be much faster than the access time to remote memories

---

Prof. Cristina Silvano, Politecnico di Milano
Distributed Shared-Mem. (DSM) Arch.

- Characterized by local memory modules distributed to the nodes, but components of a single global address space, i.e. any processor can access the local memory of any other processor.
- Whenever a processor addresses the module of the shared-memory on the local node, the access is much faster than the access to a memory module placed on a remote node.
- Larger coherent caches reduce the time spent on remote load operations.
- Directory-based protocols to support cache coherence
Large-Scale MP Designs

- DSM architectures with *non uniform memory access time (NUMA)* and *scalable interconnect*.

- Interconnection network provides low latency and high reliability.
Large Scale MP Designs

- An example of Cluster Architecture: NUMA (Non Uniform Memory Access) architecture based on Scalable Coherent Interface (SCI) to connect two SMP systems.
Multicore DSM

- Memory physically distributed among processors (rather than centralized)
- Non-uniform memory access/latency (NUMA)
- Have a larger processor count (from 8 to 32)
- Processors connected via direct (switched) and non-direct (multi-hop) interconnection networks
- Typical multicore multiprocessor chip with memory and I/O attached and an interface to an interconnection network that connects all the nodes
Clusters made by individual computers connected by an interconnection network (Message passing architectures)
Challenges of Parallel Processing

- Limited intrinsic parallelism available in programs
  - Programmer or compiler help, to expose more parallelism
- Relatively high cost of communication
  - A memory access will cost:
    - 35 to 50 cycles between cores in the same chip
    - 100 to 500+ cycles between cores in different chip
  - It’s fundamental to have cache memories
Importance of caches in MPs

- Caches reduce average access latency
- Caches reduce contention
- Because, if several cores want to access the same data, each core will access it from its own cache
- When shared data are cached, the shared values may be replicated in multiple caches
- But the use of multiple copies of the same data introduces a problem: cache coherence
Cache Coherence

- Processors may see different values through their caches:

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Cache contents for processor A</th>
<th>Cache contents for processor B</th>
<th>Memory contents for location X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Processor A reads X</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Processor B reads X</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Processor A stores 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>into X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Enforcing Coherence

- Coherent caches provide:
  - *Migration*: movement of data (faster access)
  - *Replication*: multiple copies of data (reduced contention)

- How to enforce coherence?
  - Through cache coherence protocols

- Key to implementing a cache coherence protocol is tracking the state of any sharing of a data block
Cache coherence protocols

Two classes of protocols:

- **Snooping protocols**: each core tracks the sharing status of each block
  - A cache controller monitors (snoops) on the bus, to see what is being requested by another cache

- **Directory-based protocols**: the sharing status of each block is kept in one location, called the directory
  - In SMPs: a single directory
  - In DSMs: multiple, distributed directories (one for each main memory)
Examples of snooping protocols

- **Write invalidate protocol**
  - A processor writing an item to memory **invalidates** all other copies
  - Most common protocol

- **Write update protocol**
  - A processor writing an item to memory **updates** all other copies
  - More expensive, less common protocol
Cache Coherence and Consistency

- Cache Coherence
  - All reads by any processor must return the most recently written value
  - Writes to the same location by any two processors are seen in the same order by all processors

- Memory Consistency
  - When a written value will be returned by a read?
  - In other words, in what order must a processor observe the data writes of another processor?
  - If a processor writes location A followed by location B, any processor that sees the new value of B must also see the new value of A
How do parallel procs coordinate?

- Synchronization mechanisms to access the common resources
- Message passing mechanisms to guarantee the communication among processors
Communication Models

- **Shared Memory**
  - Processors communicate with shared address space
  - Easy on small-scale machines
  - Advantages:
    - Model of choice for uniprocessors and small-scale MPs
    - Ease of programming
    - Lower latency
    - Easier to use hardware controlled caching

- **Message Passing**
  - Processors have private memories, communicate via messages
  - Advantages:
    - Less hardware, easier to design
    - Focuses attention on costly non-local operations

- **Data Parallel Model**
Data Parallel Model

- Operations can be performed in parallel on each element of a large regular data structure, such as an array.
- 1 Control Processor broadcast to many PEs
  - When computers were large, could amortize the control portion of many replicated PEs.
- Condition flag per PE so that can skip.
- Data distributed in each memory.
- Early 1980s VLSI => SIMD (Single Instruction Multiple data) rebirth.
- Data parallel programming languages lay out data to processor.
Data Parallel Model

- SIMD led to Data Parallel Programming languages
- SIMD programming model led to **Single Program Multiple Data (SPMD)** model
  - All processors execute identical program
- Data parallel programming languages still useful, do communication all at once:
  - “Bulk Synchronous” phases in which all communicate after a global barrier
Convergence in Parallel Architecture

- Complete computers connected to scalable network via communication assist
- Different programming models place different requirements on communication assist
  - **Shared address space**: tight integration with memory to capture memory events that interact with others + to accept requests from other nodes
  - **Message passing**: send messages quickly and respond to incoming messages: tag match, allocate buffer, transfer data, wait for receive posting
  - **Data Parallel**: fast global synchronization
## Characteristics of four commercial multicores

<table>
<thead>
<tr>
<th>Feature</th>
<th>AMD Opteron 8439</th>
<th>IBM Power 7</th>
<th>Intel Xenon 7560</th>
<th>Sun T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistors</td>
<td>904M</td>
<td>1200 M</td>
<td>2300 M</td>
<td>500 M</td>
</tr>
<tr>
<td>Nominal Power</td>
<td>137 W</td>
<td>140 W</td>
<td>130 W</td>
<td>95 W</td>
</tr>
<tr>
<td>Max cores/chip</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Multithreading</td>
<td>No</td>
<td>SMT</td>
<td>SMT</td>
<td>Fine-grained</td>
</tr>
<tr>
<td>Threads/core</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Clock rate</td>
<td>2.8 GHz</td>
<td>4.1 GHz</td>
<td>2.7 GHz</td>
<td>1.6 GHz</td>
</tr>
<tr>
<td>Multicore coherence</td>
<td>Snooping</td>
<td>Directory</td>
<td>Directory</td>
<td>Directory</td>
</tr>
</tbody>
</table>
Multicores

- Cores have become the new building block of chips
- Intel, with a single architecture design (the Nehalem core) has created all of its multiprocessor platforms:
  - Xeon, i3, i5, i7 are all based on the same core