Parallel Hardware

Parallel Computing – COSC 3P93

Course Outline

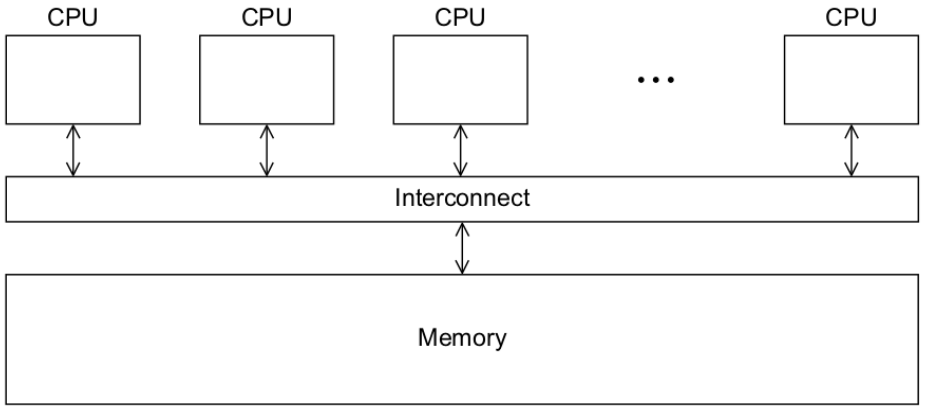
- Introduction
- Parallel Hardware
- Parallel Software
- Performance
- Parallel Programming Models
- Patterns
- SMP with Threads and OpenMP
- Distributed-Memory Programming with MPI
- Scala
- Algorithms
- Tools
- Parallel Program Design

Today Class

- Parallel Hardware
  - MIMD
    - Shared Memory

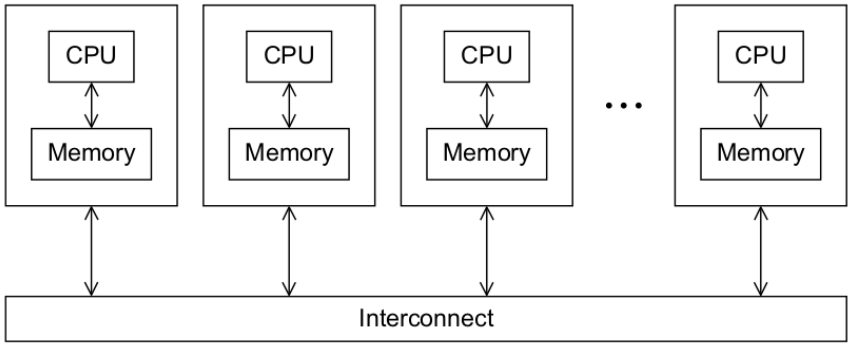
MIMD Systems

- Two types of MIMD systems → classification based on memory arrangement
  - **Shared-memory systems**
    - Autonomous processors connected to a memory system
      - Interconnection network
    - Each processor accessing each memory location
    - Processor communication → accessing shared data structures



MIMD Systems

- Two types of MIMD systems → classification based on memory arrangement



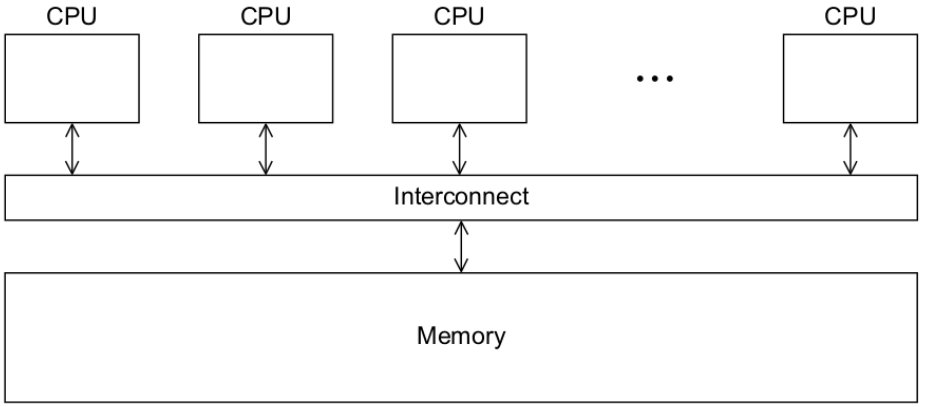
  - **Distributed-memory systems**
    - Each processor paired with its private memory
      - Processor-memory pairs communication
        - Interconnection network
    - Processor communication
      - Message passing
      - Special functions grating access to other memory regions

Shared-memory Systems

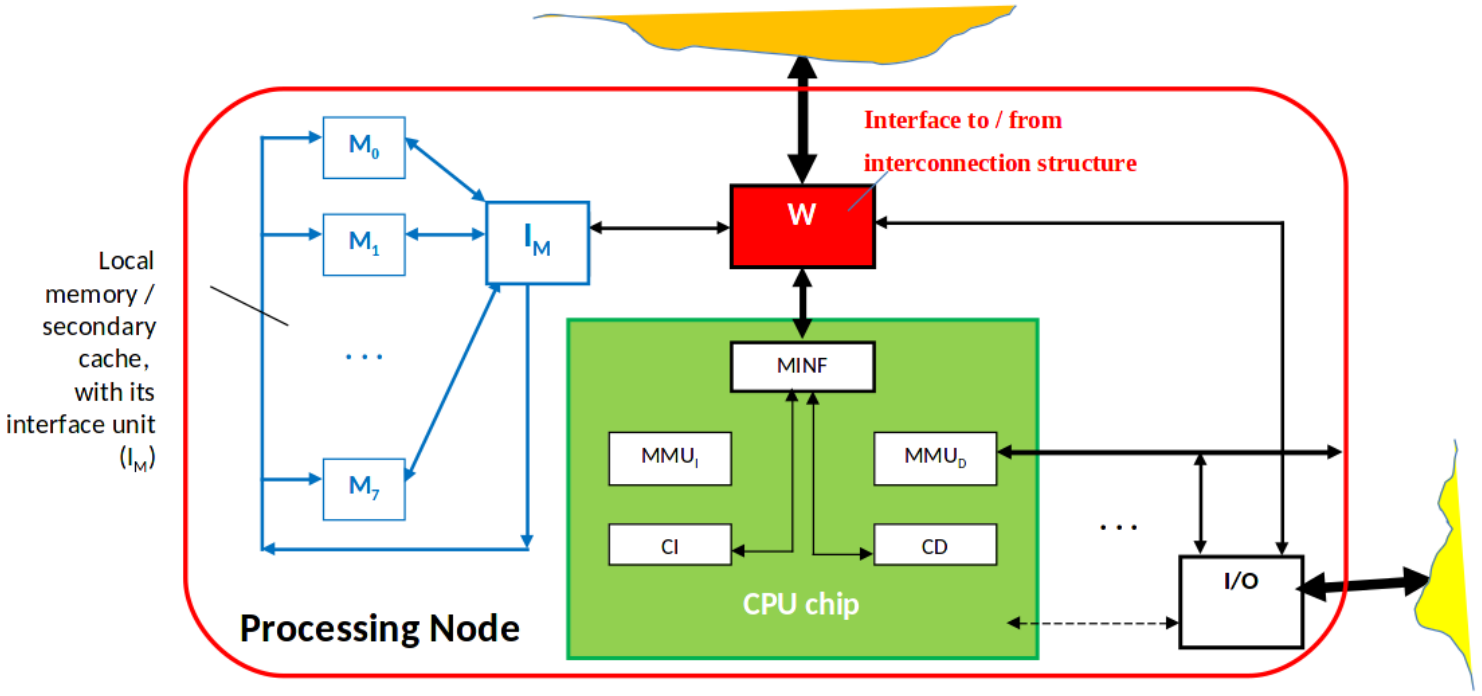
- **Shared memory parallel architecture = Multiprocessor**
  - MIMD gives a general-purpose architecture
  - Keep in mind
    - Large diffusion for high-performance servers, medium/high-end workstations
- Homogeneous: **n** identical processors
  - In general, n processing nodes
    - **(CPU, local memory/caches, local I/O)**
- Processing nodes **share** the Main Memory physical space
  - At the firmware level
    - Any processor can access any location of Main Memory
  - In presence of a memory hierarchy
    - Some Cache levels (notably, Secondary, or Tertiary Cache) can be shared
  - Shared information are allocated in shared memory supports, as well as private information

Shared-memory Systems

- Memory is common to all processors
  - Processors easily communicate by means of shared variables
- **Multicore** systems → one or more cores
  - Most widely available shared-memory systems
  - Multiple CPUs in a single chip
    - Private level 1 caches
    - Other level caches may be shared



Shared-memory Processing Nodes



Example

- Interface unit W in charge of
  - Performing cache block transfers (or other memory accesses)
  - Masking to the CPU the structure of the shared memory and communication network
- The firmware protocol implemented in the CPU to request a block transfer (memory access) **cannot** be modified with the uniprocessor architecture
- Unit W adapts this firmware protocol to the features of
  - The shared memory
    - Ex → individuates the referred memory module, and the path to reach it
  - The features of the communication network
    - Ex → inserts the CPU request into a message with proper format (source, destination node identifier, information about the routing and flow control strategy, etc) and size (one or more packets, according to the link width of the interconnection network)

Example (cont...)

- Though the main mode for nodes cooperation is shared memory
  - Some cooperation actions can be done also through **direct communications by value**
    - The **I/O subsystem** must be used
    - An I/O unit is in charge of interfacing the direct communications between nodes
      - Adapting the CPU request to the rest of the systems
    - **The same interconnection structure** for shared memory is used for direct communications too
    - In the figure
      - This unit exploits also the DMA mode inside the processing node

Physical Addressing Spaces

- Multiprocessor memory as a whole has high capacity
  - Expandable to Tera-words or more
  - Physical address of 40 bits or more
- CPU must be able to generate such physical addresses
  - An impact on the design of CPU
  - Not the only one impact case (other meaningful cases will be met)
  - Standard CPUs and other structures can be adopted
    - But, they are prone to some multiprocessor requirements and peculiarities
      - Ex
        - Support of indivisible sequences of memory accesses
        - Cache-coherence
- Note
  - There is no a-priori relation between the size of physical addresses and of logical addresses

Classes of Multiprocessor Architectures

- Two distinctions according to
  - **Processes-to-processors mapping**
    - Dynamic or static correspondence between processes and processors
    - **Anonymous** vs **dedicated** processors
  - **Organization of modular memory as “seen” by the processors**
    - Uniform or non uniform access time to parts (modules) of the shared memory
    - Uniform memory access (**SMP** or **UMA**) vs non-uniform memory access (**NUMA**)

Process-to-processor Mapping

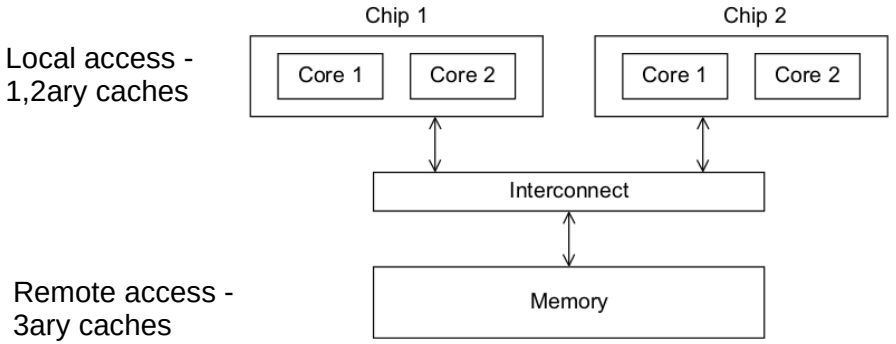
- **Anonymous processors** architecture
  - Any process can be executed by any processor
  - When a waiting process is woken-up
    - Its execution is resumed on a different processor
  - **Dynamic** mapping as with the low-level scheduling funct.
  - A unique **Ready List** exists for all processors
    - Shared by all processes
- **Dedicated processors** architecture
  - Static association of disjoint subsets of active processes to processors
  - Decided at loading time
  - Multiprogrammed nodes
  - Each node has its own **Ready List**
    - Not shared by processes allocated to other nodes
  - **Re-allocation** of processes to nodes can be done sporadically
    - For fault-tolerance or load-balancing
      - Conceptually → considered static mapping

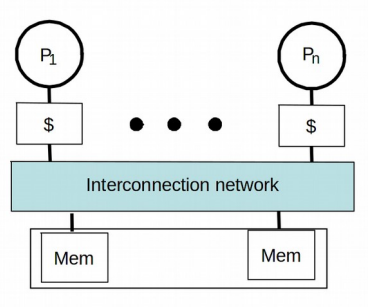
Shared-memory Systems

- **Multiple multicore systems**
  - **Type 1**
    - All processors directly connected with main memory
  - **Type 2**
    - Each processor directly connected with a block of main memory
      - Processors accessing each other’s blocks of memory through special hardware

Shared-memory Systems Type 1

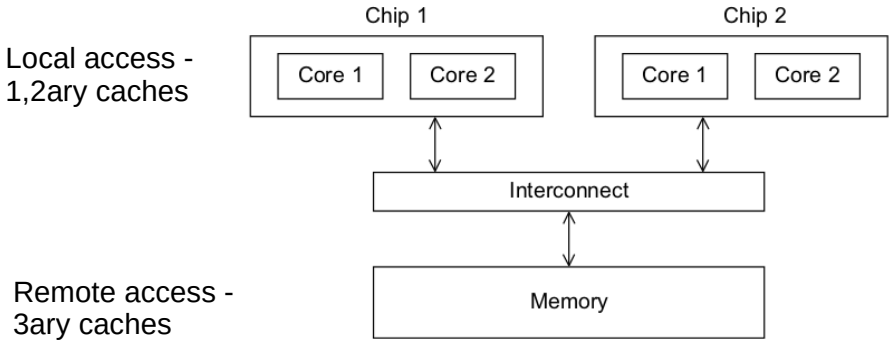
- **UMA → Uniform Memory Access**
  - Same memory access time to all processors
    - Same location to all cores
    - The base memory access latency of processor **i** to memory **j**
      - Measured in absence of conflicts
      - Constant and independent of **i** and **j**

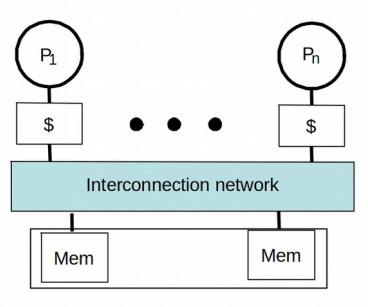




Uniform Memory Access

- In Symmetric MultiProcessor (**SMP**) Architectures
- Tightly-coupled systems
- High degree of resource sharing
- Suitable for general-purpose and time-sharing applications by multiple users
- Easier to program
  - No worries about memory access times
- Example
  - 2ary caches are private to nodes
  - 3ary, or main memory, is shared



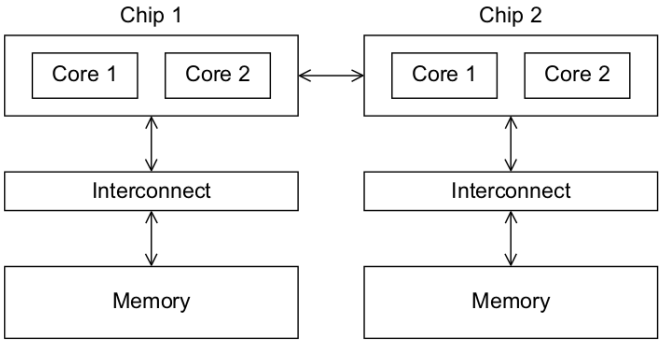


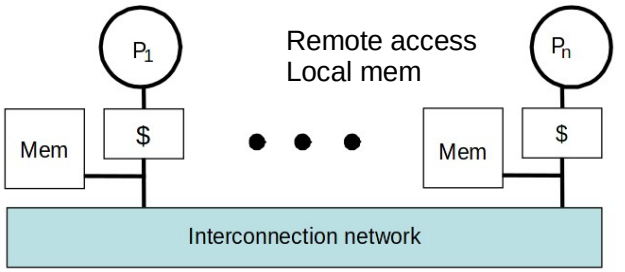
UMA – Symmetry

- Symmetric and asymmetric multiprocessors
  - **Symmetric**
    - All processors have equal access to all peripheral devices
    - All processors are identical
  - **Asymmetric**
    - One processor (master) executes the operating system
    - Other processors may be of different types and may be dedicated to special tasks

Shared-memory Systems Type 2

- **NUMA → NonUniform Memory Access**
  - Faster access to directly connected memory than memory through other chips
    - The access time varies with the location of the memory word
    - Shared memory is distributed to local memories
    - All local memories form a global address space accessible by all processors
  - Different access times
    - Cache, Local memory, Remote memory

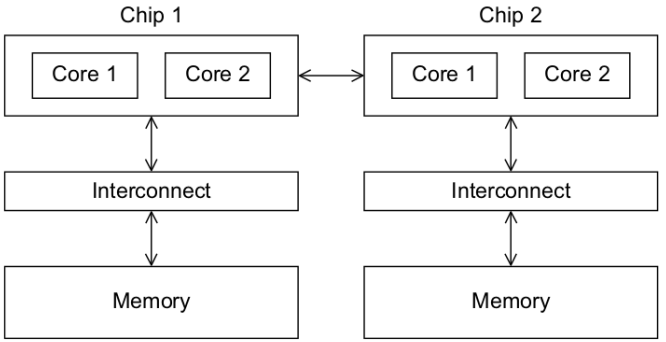


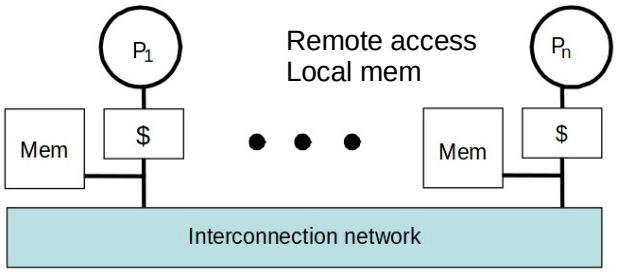


Local access - 1,2ary caches

NonUniform Memory Access

- Variant
  - COMA → Cache-only Memory Architecture
- Great potential for using larger amounts of memory
- Example
  - 2ary caches are private of the nodes
  - Local memories of nodes are all shared
  - Every local memory can be interleaved (the shared memory, as a whole, is not) or long-word





Local access -
1,2ary caches

Shared-memory Architecture Combinations

- Processor mapping and shared-memory strategies → feasible architectures
  - Anonymous processors + UMA
  - Anonymous processors + NUMA
  - Dedicated processors + UMA
  - Dedicated processors + NUMA
- Most “natural” (most popular) combinations: **1** and **4**
- Food for thought
  - What is the role of memory hierarchies in such architectures?
    - Memory hierarchies tend to smooth the difference between architectural combinations 1 and 2 (anonymous processors + UMA, anonymous processors + NUMA) and between 3 and 4 (dedicated processors + UMA, dedicated processors + NUMA)
    - Although combinations 1, 4 appear “natural”, the proper use of memory hierarchies could render the other combinations acceptable as well

Issues in Shared-Memory

- Issues
  - Access latency
  - Memory conflicts
- In multiprocessors
  - The proper use of **local memories** and **memory hierarchies**
    - Main issue for performance optimization
- Target
  - To minimize access latency
  - To minimize memory conflicts

Minimizing Access Latency

- Interconnection networks latency is dependent of the number of nodes
  - Linear (bus)
  - Logarithmic (butterflies, high dimension cubes, trees)
  - Square-root (low dimension cubes)
- Shared memory access are “expensive”
  - In **UMA** all accesses to shared memory are “remote”
  - In **NUMA** some accesses to shared memory are “remote” and other ones are local

Minimizing Access Latency

- **UMA** goal → try to dynamically allocate useful information in local caches (C1, C2), as usually
- **NUMA** goal → try to statically allocate useful information in local memories and dynamically allocate local memory information in local caches
  - In a **dedicated processors** architecture, all the private information of processes mapped to the a node are allocated in the local memory of such node (**code**, **data**), as well as some shared information
  - **Remote accesses** only for (the remaining) shared information

Minimizing Shared Memory Conflicts

- The system can be modeled as a **queueing system**
  - Processing nodes are clients
  - Shared memory modules are servers, including the interconnection structure
  - For instance → M/D/1 queue for each memory module
- The mem. access latency is the server **Response Time**
- Critical dependency on the server utilization factor
  - A measure of memory modules congestion
  - Or a measure of processor “conflicts” to access the same memory module
- The **interarrival time** has to be taken as high as possible
  - The importance of **local** accesses
  - Thus the importance of the best exploitation of **local memories** (NUMA) and **caches** (SMP)

Issues in Shared-Memory

- Issues
  - Access latency
  - Memory conflicts
- In multiprocessors
  - The proper use of **local memories** and **memory hierarchies**
    - Main issue for performance optimization
- Target
  - To minimize access latency
  - To minimize memory conflicts
- **Solutions → drawback**
  - Existence of of writable shared information in caches introduces the problem of cache coherence
    - How to guarantee that the contents of distinct caches are consistent

Class Recap

- Parallel Hardware
  - MIMD
    - Shared Memory

Next Class

- Parallel Hardware
  - MIMD
    - Distributed Memory
    - Interconnection networks
    - Cache coherence
    - Shared-memory versus distributed-memory