LAU
الجامعة اللبنانية الأميركية
Lebanese American University

CSC 447: Parallel Programming for Multi-Core and Cluster Systems

Parallel Architectures

Instructor: Haidar M. Harmanani

Spring 2021

Today's Agenda

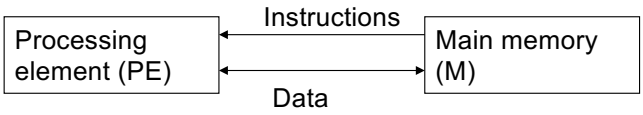
- Flynn's taxonomy
- Classification based on the memory arrangement
- Classification based on communication
- Classification based on the kind of parallelism
  - Data-parallel
  - Function-parallel

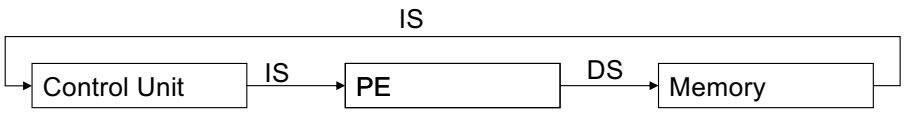
Flynn's Taxonomy

- The most universally excepted method of classifying computer systems (1966)
- Any computer can be placed in one of 4 broad categories
  - SISD: Single instruction stream, single data stream
  - SIMD: Single instruction stream, multiple data streams
  - MIMD: Multiple instruction streams, multiple data streams
  - MISD: Multiple instruction streams, single data stream

SISD

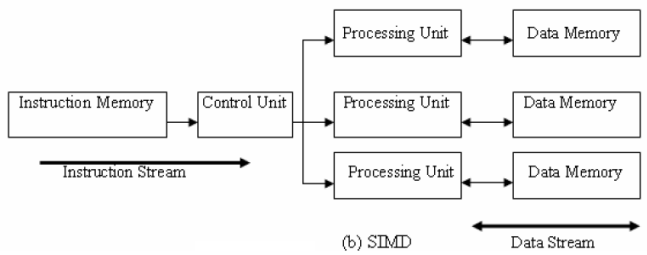
- At one time, one instruction operates on one data
- Traditional sequential architecture





SIMD

- At any given time, one instruction operates on many data
  - Data parallel architecture
  - Vector architecture has similar characteristics but achieve the parallelism with pipelining.
- Array processors

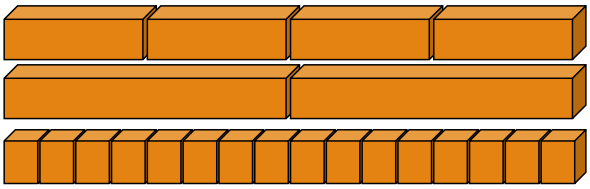


SIMD Architectures

- Fine-grained
  - Image processing application
  - Large number of PEs
  - Minimum complexity PEs
  - Programming language is a simple extension of a sequential language
- Coarse-grained
  - Each PE is of higher complexity and it is usually built with commercial devices
  - Each PE has local memory

SSE / SSE2 SIMD on Intel

- SSE2 data types: anything that fits into 16 bytes, e.g.,



4x floats

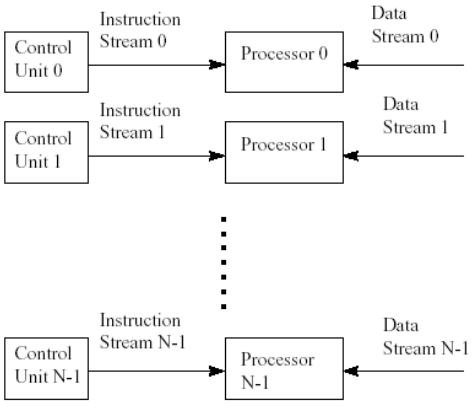
2x doubles

16x bytes

- Instructions perform add, multiply etc. on all the data in this 16-byte register in parallel
- Challenges:
  - Need to be contiguous in memory and aligned
  - Some instructions to move data around from one part of register to another
- Similar on GPUs, vector processors (but many more simultaneous operations)

MIMD

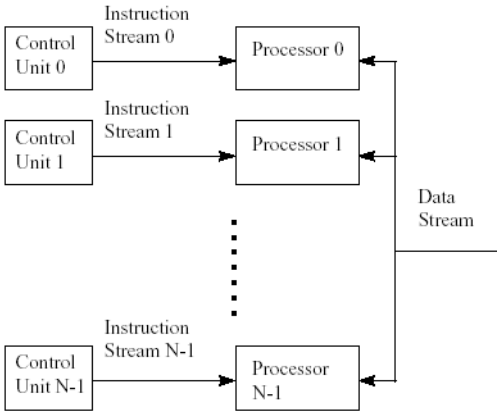
- Multiple instruction streams operating on multiple data streams
  - Classical distributed memory or SMP architectures



MIMD system architecture of [Fly66]

MISD

- Not commonly seen.
- Systolic array is one example of an MISD architecture.



MISD system architecture of [Fly66]

Flynn Taxonomy

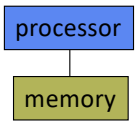
- Advantages
  - Universally accepted
  - Compact Notation
  - Easy to classify a system
- Disadvantages
  - Very coarse-grain differentiation among machine systems
  - Comparison of different systems is limited
  - Interconnections, I/O, memory not considered in the scheme

Parallel Architecture Types

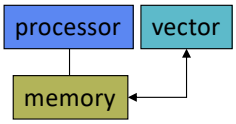
Parallel Architecture Types

Uniprocessor

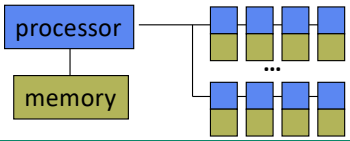
- Scalar processor



- Vector processor

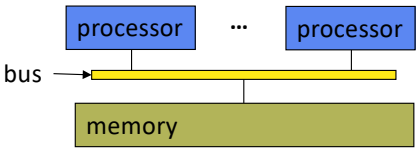


- Single Instruction Multiple Data (SIMD)

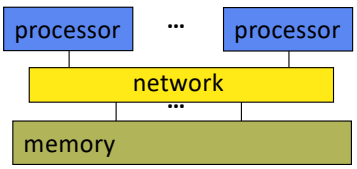


Shared Memory Multiprocessor (SMP)

- Shared memory address space
- Bus-based memory system



- Interconnection network



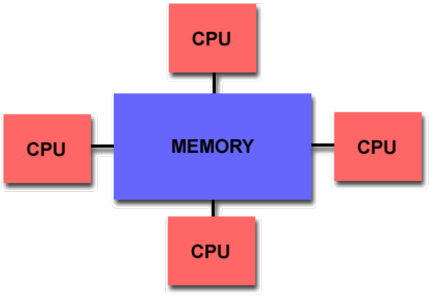
Shared-Memory Multiprocessors

- Uniform Memory Access (UMA)
- Non-Uniform Memory Access (NUMA)
- Cache-only Memory Architecture (COMA)

- Memory is common to all the processors.
- Processors easily communicate by means of shared variables.

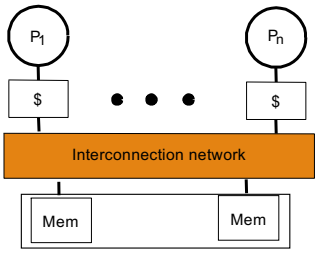
Shared-Memory Multiprocessors

- Shared memory parallel computers vary widely, but generally have in common the ability for all processors to access all memory as global address space.
- Multiple processors can operate independently but share the same memory resources.
- Changes in a memory location effected by one processor are visible to all other processors.
- Shared memory machines can be divided into two main classes based upon memory access times: UMA and NUMA.



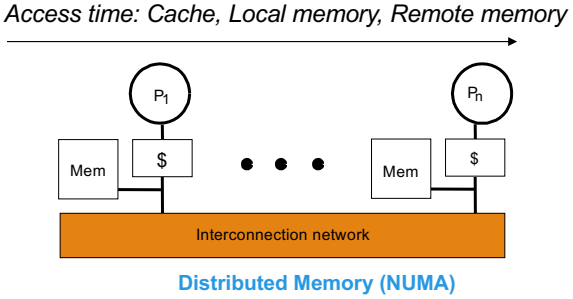
The UMA Model

- Tightly-coupled systems (high degree of resource sharing)
- Suitable for general-purpose and time-sharing applications by multiple users.



The NUMA Model

- The access time varies with the location of the memory word.
- Shared memory is distributed to local memories.
- COMA - Cache-only Memory Architecture
- All local memories form a global address space accessible by all processors



Distributed Memory (NUMA)

Shared Memory : UMA vs. NUMA

- Uniform Memory Access (UMA):
  - Most commonly represented today by Symmetric Multiprocessor (SMP) machines
  - Identical processors
  - Equal access and access times to memory
  - Sometimes called CC-UMA - Cache Coherent UMA. Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.
- Non-Uniform Memory Access (NUMA):
  - Often made by physically linking two or more SMPs
  - One SMP can directly access memory of another SMP
  - Not all processors have equal access time to all memories
  - Memory access across link is slower
  - If cache coherency is maintained, then may also be called CC-NUMA - Cache Coherent NUMA

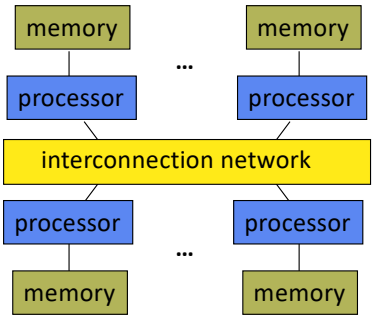
Shared Memory: Pro and Con

- Advantages
  - Global address space provides a user-friendly programming perspective to memory
  - Data sharing between tasks is both fast and uniform due to the proximity of memory to CPUs
- Disadvantages:
  - Primary disadvantage is the lack of scalability between memory and CPUs. Adding more CPUs can geometrically increase traffic on the shared memory-CPU path, and for cache coherent systems, geometrically increase traffic associated with cache/memory management.
  - Programmer responsibility for synchronization constructs that insure "correct" access of global memory.
  - Expense: it becomes increasingly difficult and expensive to design and produce shared memory machines with ever increasing numbers of processors.

Parallel Architecture Types

Distributed Memory Multiprocessor

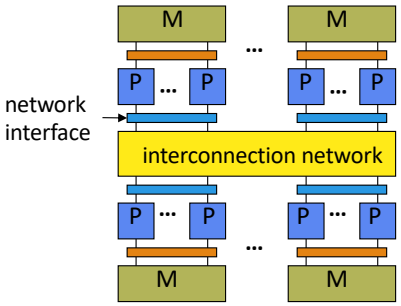
- Message passing between nodes



- Massively Parallel Processor (MPP)
  - Many, many processors

Cluster of SMPs

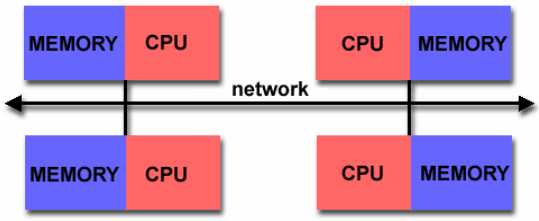
- Shared memory addressing within SMP node
- Message passing between SMP nodes



- Can also be regarded as MPP if processor number is large

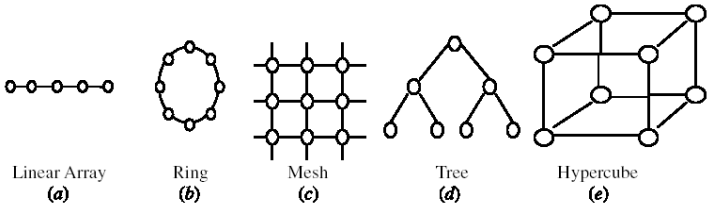
Distributed Memory

- Distributed memory systems vary widely but share a common characteristic
  - Require a communication network to connect inter-processor memory.
- Processors have their own local memory.
  - No concept of global address space across all processors.
- Each processor operates independently of other processors
  - Hence, the concept of cache coherency does not apply.
- When a processor needs access to data in another processor, it is usually communicated via messaging
  - Communication and Synchronization are the programmer's responsibility!
- The network "fabric" used for data transfer varies widely

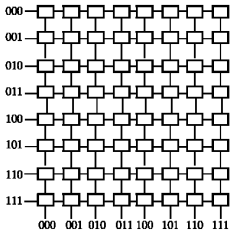


Interconnection Networks

- Static networks



- Dynamic networks

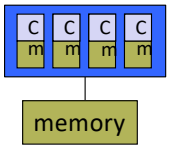


Distributed Memory: Pros and Cons

- Advantages
  - Memory is scalable with number of processors. Increase the number of processors and the size of memory increases proportionately.
  - Each processor can rapidly access its own memory without interference and without the overhead incurred with trying to maintain cache coherency.
  - Cost effectiveness: can use commodity, off-the-shelf processors and networking.
- Disadvantages
  - The programmer is responsible for many of the details associated with data communication between processors.
  - It may be difficult to map existing data structures, based on global memory, to this memory organization.
  - Non-uniform memory access (NUMA) times

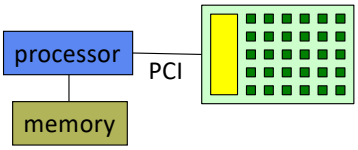
Parallel Architecture Types

- **Multicore**
  - Multicore processor

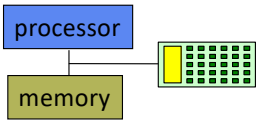


cores can be hardware multithreaded (hyperthread)

  - GPU accelerator

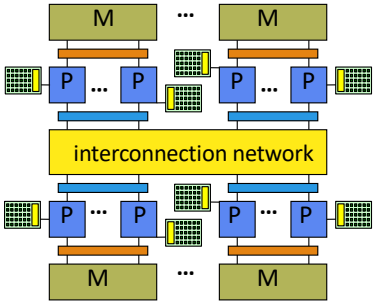


  - "Fused" processor accelerator



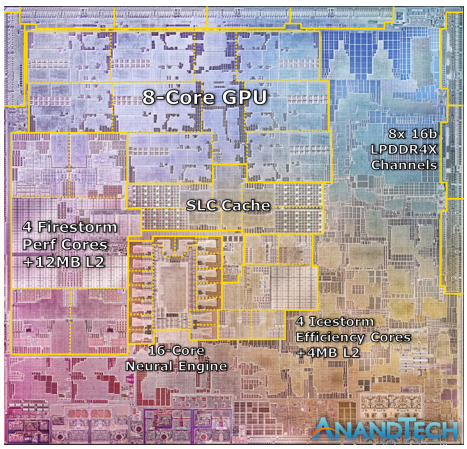
Multicore SMP+GPU Cluster

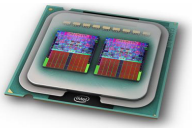
- Shared memory addressing within SMP node
- Message passing between SMP nodes
- GPU accelerators attached

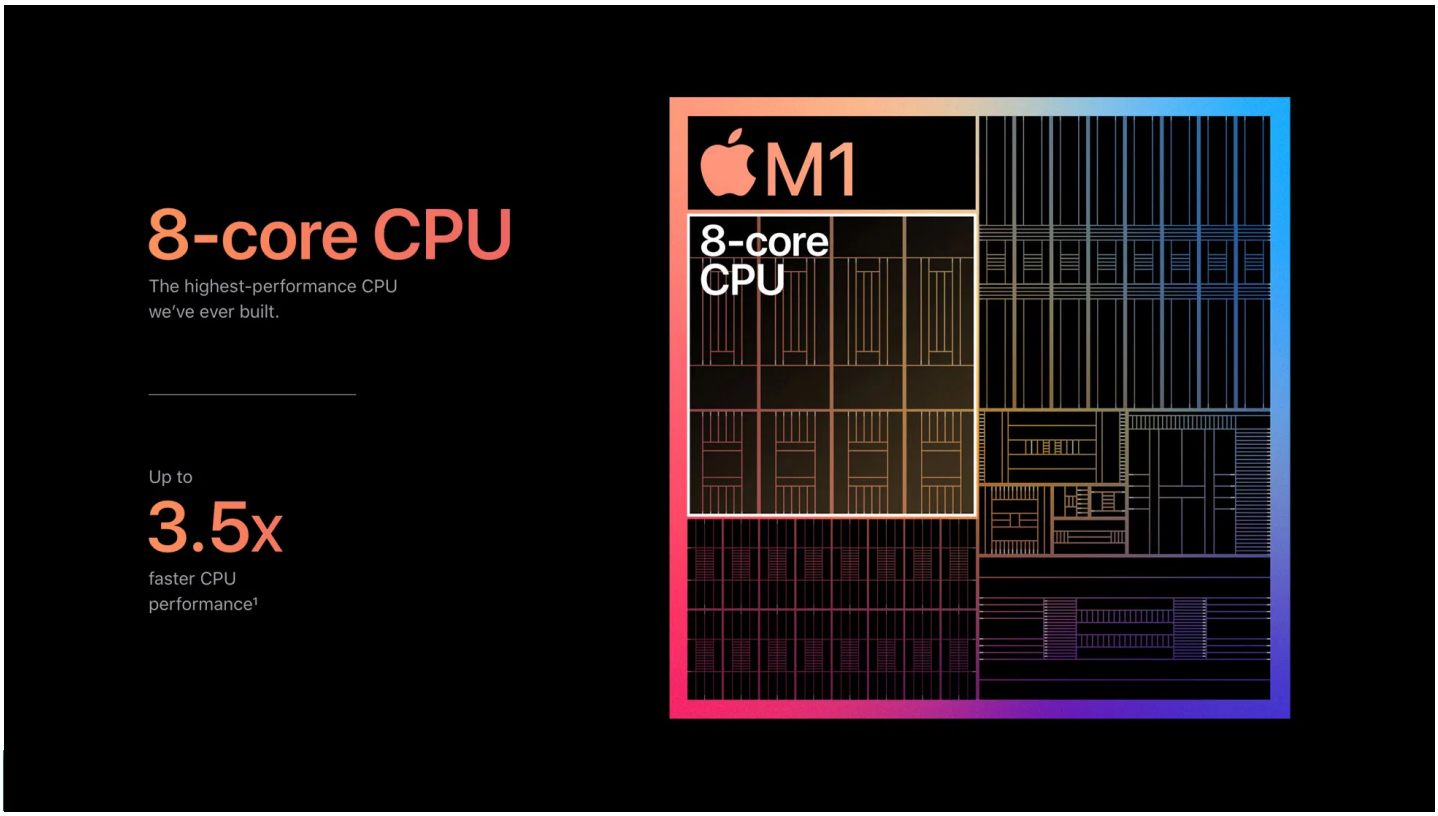


Multi-core Processors

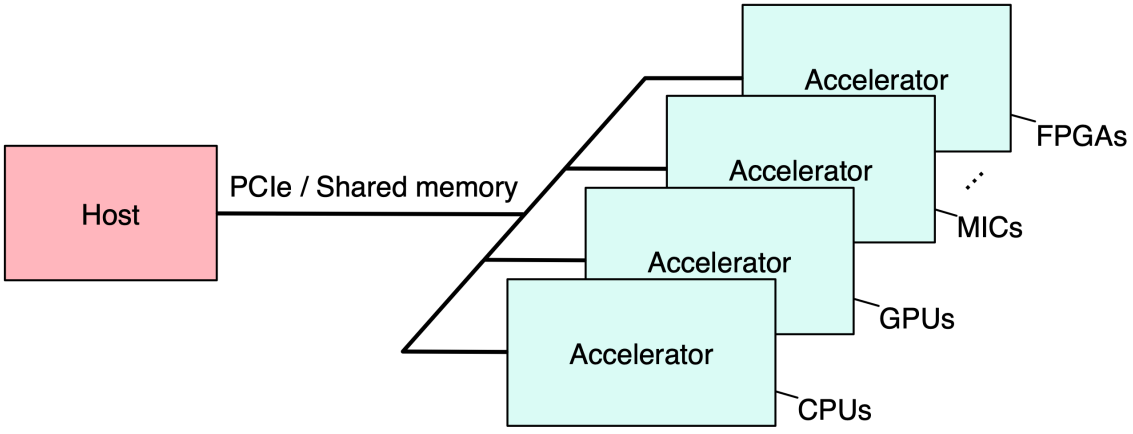
- All cores which exist in a die are exactly identical







Accelerators



GPU Accelerators