Taxonomy of Parallel Processors

Processor Organizations

- Single instruction, single data stream (SISD)
  - Uniprocessor
  - Vector Processor

- Single instruction, multiple data stream (SIMD)
  - Array Processor

- Multiple instruction, single data stream (MISD)

- Multiple instruction, multiple data stream (MIMD)
  - Shared Memory (Tightly Coupled)
  - Distributed Memory (Loosely Coupled)
    - Symmetric Multiprocessor (SMP)
    - Nonuniformed Memory Access (NUMA)
    - Cluster
Taxonomy of Parallel Processors

Processor Organizations

Single instruction, single data stream (SISD)

Uniprocessor
Parallel Processors

• Single instruction, single data stream – SISD
  ‣ Single processor
  ‣ Single instruction stream
  ‣ Data stored in single memory
Taxonomy of Parallel Processors

Processor Organizations

- Single instruction, single data stream (SISD)
  - Uniprocessor
  - Vector Processor
  - Array Processor
- Single instruction, multiple data stream (SIMD)
Parallel Processors

• Single instruction, multiple data stream – SIMD

• Single machine instruction
  ‣ Each instruction executed on different set of data by different processors

• Number of processing elements
  ‣ Machine controls simultaneous execution
    • Lockstep basis
  ‣ Each processing element has associated data memory

• Application: Vector and array processing
Parallel Processors

- Single instruction, multiple data stream – SIMD

Diagram showing the flow of instructions and data streams.
Taxonomy of Parallel Processors

Processor Organizations

- Single instruction, single data stream (SISD)
  - Uniprocessor
  - Vector Processor

- Single instruction multiple data stream (SIMD)
  - Array Processor

- Multiple instruction, single data stream (MISD)
Parallel Processors

• Multiple instruction, single data stream – MISD
  ‣ Sequence of data
  ‣ Transmitted to set of processors
  ‣ Each processor executes different instruction sequence

• Do not know any implemented case
Parallel Processors

- Multiple instruction, single data stream – MISD
Taxonomy of Parallel Processors

Processor Organizations

- **Single instruction, single data stream (SISD)**
  - **Uniprocessor**
  - **Vector Processor**

- **Single instruction multiple data stream (SIMD)**
  - **Array Processor**

- **Multiple instruction, single data stream (MISD)**

- **Multiple instruction, multiple data stream (MIMD)**
  - **Shared Memory (Tightly Coupled)**
  - **Distributed Memory (Loosely Coupled)**
    - **Symmetric Multiprocessor (SMP)**
    - **Nonuniformed Memory Access (NUMA)**
    - **Cluster**
Parallel Processors

• Multiple instruction, multiple data stream- MIMD
  ‣ Set of processors
  ‣ Simultaneously executes different instruction sequences
  ‣ Different sets of data

• Examples: SMPs, NUMA systems, and Clusters
Parallel Processors

- Multiple instruction, multiple data stream - MIMD
Taxonomy of Parallel Processors

Processor Organizations

- Single instruction, single data stream (SISD)
  - Uniprocessor
  - Vector Processor

- Single instruction, multiple data stream (SIMD)
  - Array Processor

- Multiple instruction, single data stream (MISD)

- Multiple instruction, multiple data stream (MIMD)
  - Shared Memory (Tightly Coupled)
  - Distributed Memory (Loosely Coupled)
    - Symmetric Multiprocessor (SMP)
    - Nonuniformed Memory Access (NUMA)
    - Cluster
Symmetric Multiprocessors (SMP)

Processor 1

Cache

Processor 2

Cache

... 

Processor N

Cache

Interconnection Network

Memory 1

Memory 2

... 

Memory M
Symmetric Multiprocessors (SMP)

- A collection of processors and a collection of memory connected through some interconnect
- Symmetric because latency for any processor to access any memory is constant – uniform memory access (UMA)
Shared Memory Architectures

- Key differentiating feature: the address space is shared, i.e., any processor can directly address any memory location and access them with load/store instructions.
Distributed Memory Multiprocessors

Memory 1
Cache
Processor 1

Memory 2
Cache
Processor 2

... 

Memory M
Cache
Processor N

Interconnection Network
Distributed Memory Multiprocessors

• Each processor has local memory that is accessible through a fast interconnect

• The different nodes are connected as I/O devices with (potentially) slower interconnect

• Local memory access is a lot faster than remote memory
  ‣ Nonuniform memory access (NUMA)
Amdahl's Law

• By Gene Amdahl

• This law answers the critical question:
  ‣ How much of a speedup one can get for a given parallelized task?

• If s is the fraction of a calculation that is sequential, and (1-s) is the fraction that can be parallelized, then the maximum speed-up that can be achieved by using n processors is
  ‣ Speed-up = \( \frac{1}{s + \frac{1}{s}} \)
Amdahl's Law

• If 80% of a calculation can be parallelized, i.e. 20% is sequential, then what is the maximum speed-up which can be achieved on 8 processors?
  ‣ What if we double the number of processors (n = 16)?
  ‣ What if we double the number of processors again (n = 32)?

• What if the number of processors is 1000?
Amdahl's Law

• If 50% of a calculation can be parallelized, i.e. 50% is sequential, then what is the maximum speed-up which can be achieved on 8 processors?
  ‣ What if we double the number of processors (n = 16)?
  ‣ What if we double the number of processors again (n = 32)?

• What if the number of processors is 1000?
Amdahl's Law

- **Key Insights**
  - The performance of any system is constrained by the speed or capacity of the slowest point.
  - The impact of an effort to improve the performance of a program is primarily constrained by the amount of time that the program spends in parts of the program NOT TARGETED by the effort.
  - Amdahl's Law is a statement of the maximum theoretical speed-up you can ever hope to achieve.
  - The actual speed-ups are always less than the speed-up predicted by Amdahl's Law.
Amdahl's Law

• For software and hardware engineers MUST have a very deep understanding of Amdahl's Law if they are to avoid having unrealistic performance expectations

1. For systems folks: this law allows you to estimate the net performance benefit a new hardware feature will add to program executions

2. For software folks: this law allows you to estimate the amount of parallelism your program/algorithm can achieve before you start writing your parallel code
Next Class

- Heterogeneous Processors