Introduction II

Overview

- Today we will introduce **multicore hardware** (we will introduce many-core hardware prior to learning OpenCL)
- We will also consider the relationship between computer hardware and programming

Benefits of Multicore Hardware

Speedup

- The goal of multiple processor is to increase performance

$$S(p) = \frac{t_s \text{ (Execution time on a single processor)}}{t_p \text{ (Execution time with } p \text{ processors)}}$$

- Linear speedup – *"a speedup factor of p with p processors"*
- Is superlinear speedup (> p) possible?
  - i.e. when $t_p < t_s/p$

Benefits of Multicore Hardware

Speedup

- The goal of multiple processor is to increase performance

$$S(p) = \frac{t_s \text{ (Execution time on a single processor)}}{t_p \text{ (Execution time with } p \text{ processors)}}$$

- Linear speedup – *"a speedup factor of p with p processors"*
- Is superlinear speedup (> p) possible?
  - i.e. when $t_p < t_s/p$ – *this would mean that the parallel parts of the program can be executed faster in sequence then t_s!*

Benefits of Multicore Hardware

Speedup

- Cases where superlinear speedup is possible:
  - When multicore system processors have more memory than single processor system
  - When hardware accelerators are used in the multiprocessor system and not available in the single processor system
  - When a nondeterministic algorithm is executed (e.g., a solution can be found quickly in one part of parallel implementation)

Parallel Architecture Taxonomy

Instruction Stream \ Data Stream	Single	Multiple
Single	SISD	SIMD
Multiple	MISD	MIMD

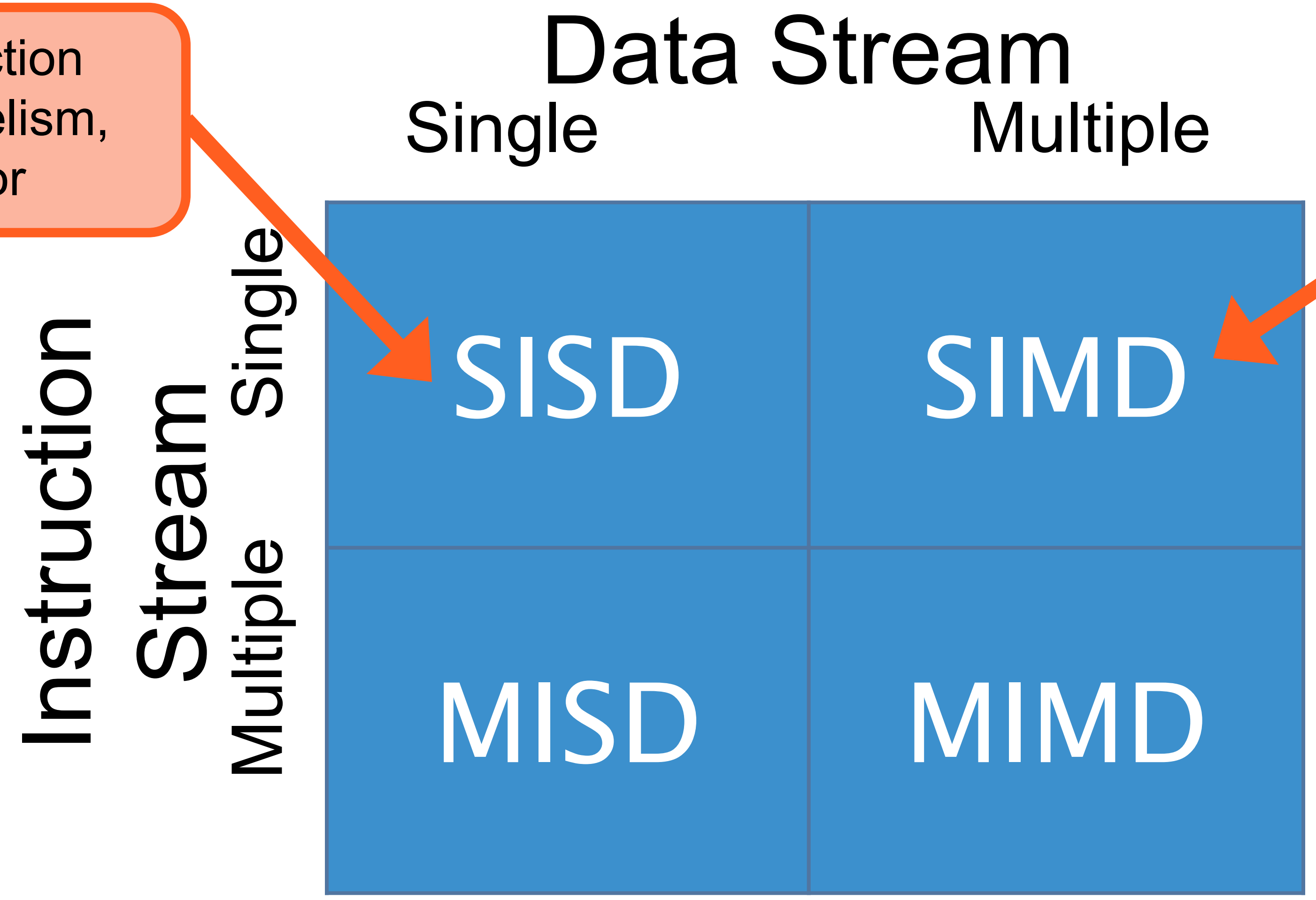
Parallel Architecture Taxonomy

Only instruction level parallelism, uniprocessor → SISD

Instruction Stream \ Data Stream	Single	Multiple
Single	SISD	SIMD
Multiple	MISD	MIMD

Parallel Architecture Taxonomy

Only instruction level parallelism, uniprocessor



Many core processors most common, **Data level parallelism** – use each instruction with multiple data elements

Parallel Architecture Taxonomy

Only instruction level parallelism, uniprocessor

Instruction Stream \ Data Stream	Single	Multiple
Single	SISD	SIMD
Multiple	MISD	MIMD

Many core processors most common, **Data level parallelism** – use each instruction with multiple data elements

Multicore processor, **Thread level parallelism** – each processor has an instruction with data

Parallel Architecture Taxonomy

Only instruction level parallelism, uniprocessor

Many core processors most common, **Data level parallelism** – use each instruction with multiple data elements

Instruction Stream \ Data Stream	Single	Multiple
Single	SISD	SIMD
Multiple	MISD	MIMD

Each processor performs different instructions on the same data – uncommon

Multicore processor, **Thread level parallelism** – each processor has an instruction with data

Parallel Architecture Taxonomy

- SIMD *vs.* MIMD
  - SIMD
    - Single Instruction Stream, Multiple Data Streams
    - Data-level parallelism can be exploited
  - MIMD
    - Multiple Instruction Streams, Multiple Data Streams
    - Thread-level parallelism can be exploited
    - Relatively low cost to build due to the use of same processors as those found in single processor machines
  - In general MIMD is more flexible than SIMD

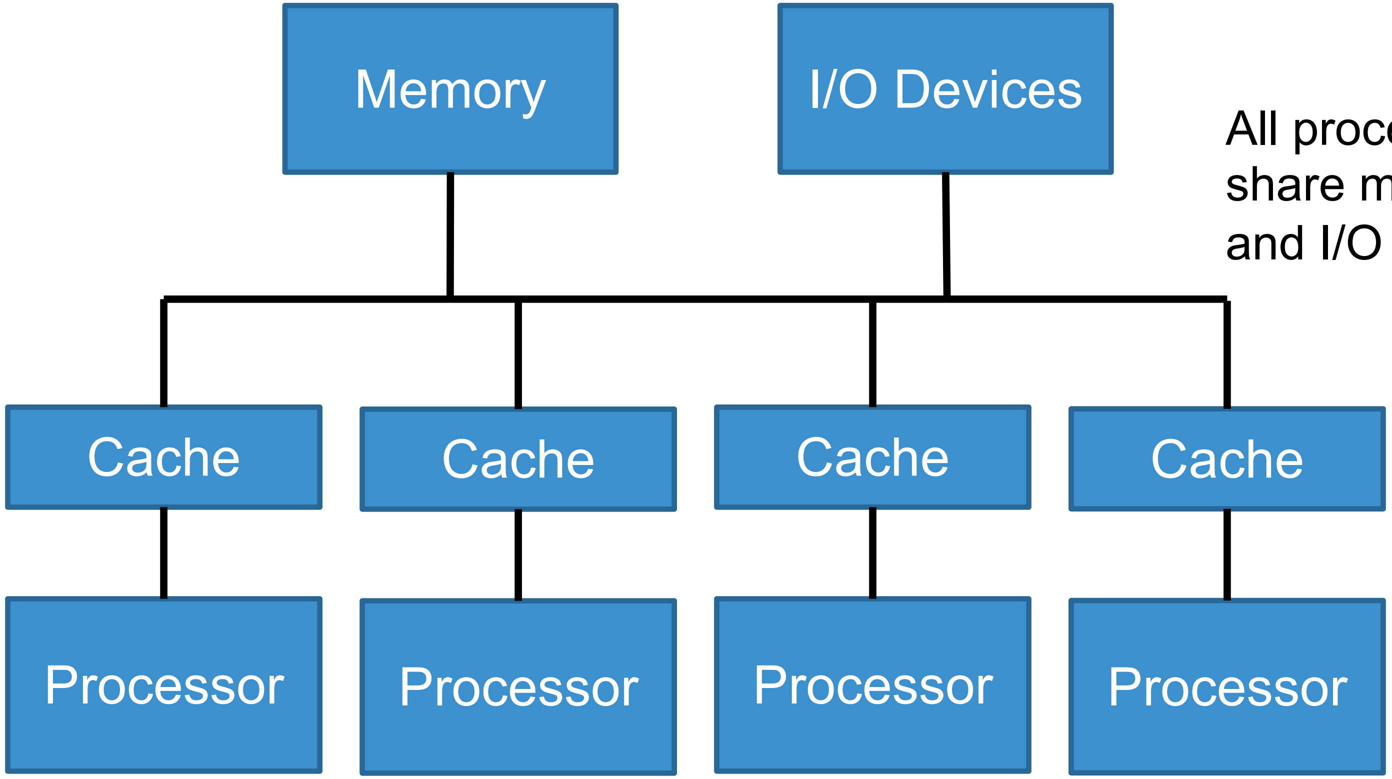
MIMD

- The flexibility of MIMD is demonstrated by the two categories of MIMDs currently used:
  1. Centralized Shared-Memory Architectures (< 100 processors)
  2. Distributed-Memory Architectures (> 100 processors)

Centralized Shared-Memory Architectures

- SMP (Symmetric Shared-Memory Multiprocessors) or NUMA (Non-Uniform Memory Access)
- **Example:** Multi-core processors
  - Multiple processors on the same die

Centralized Shared-Memory Architectures



All processors share memory and I/O devices

Distributed-Memory Architectures

- Two important aspects of these architectures is the processors and the interconnection network
- **Example:** Clusters

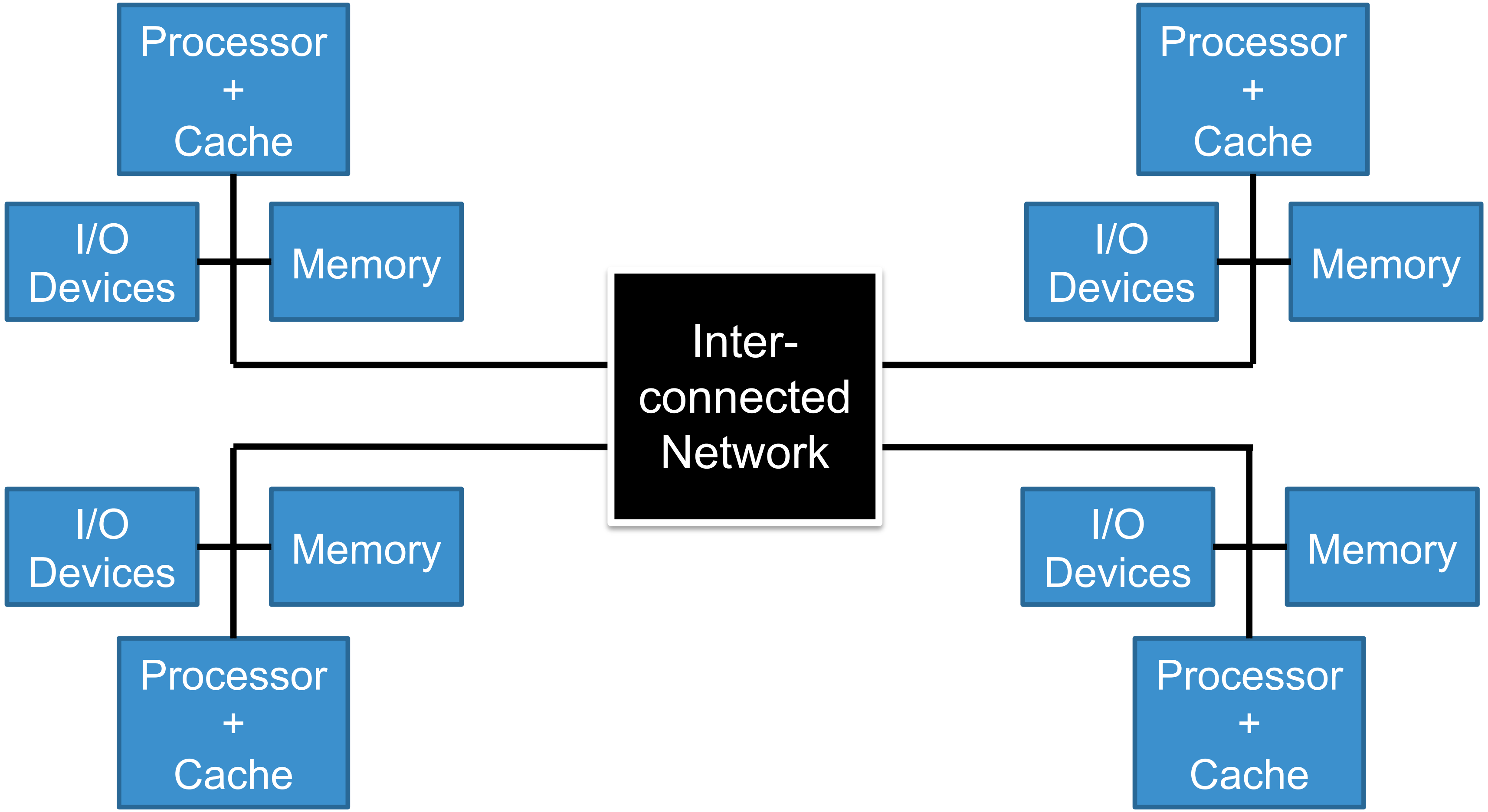
Distributed-Memory Architectures

- Can have a shared memory address space or multiple address spaces
- If shared memory address space

  …communicate used load and store instructions.
- If multiple address spaces

  …communicate via message-passing
  - Message Passing Interface (MPI) library used in C (and other languages)

Distributed-Memory Architectures



How do we take advantage of MIMD?

- Multiple processes (programs) executing at the same time
- A single program with multiple threads executing at the same time
  - Many general-purpose programming languages support multi-thread concurrent programs!
  - **Example:** Java, C++

Software Concurrency

- Hardware improvements *can* have an affect on how we develop software
- Instruction level parallelism is typically independent of whether or not software is sequential or concurrent
- Thread level parallelism techniques like multicore are usually dependent on the software being concurrent!

Instruction-Level vs. Thread-Level Parallelism

A program can contain multiple threads

↓

Each thread contains many instructions

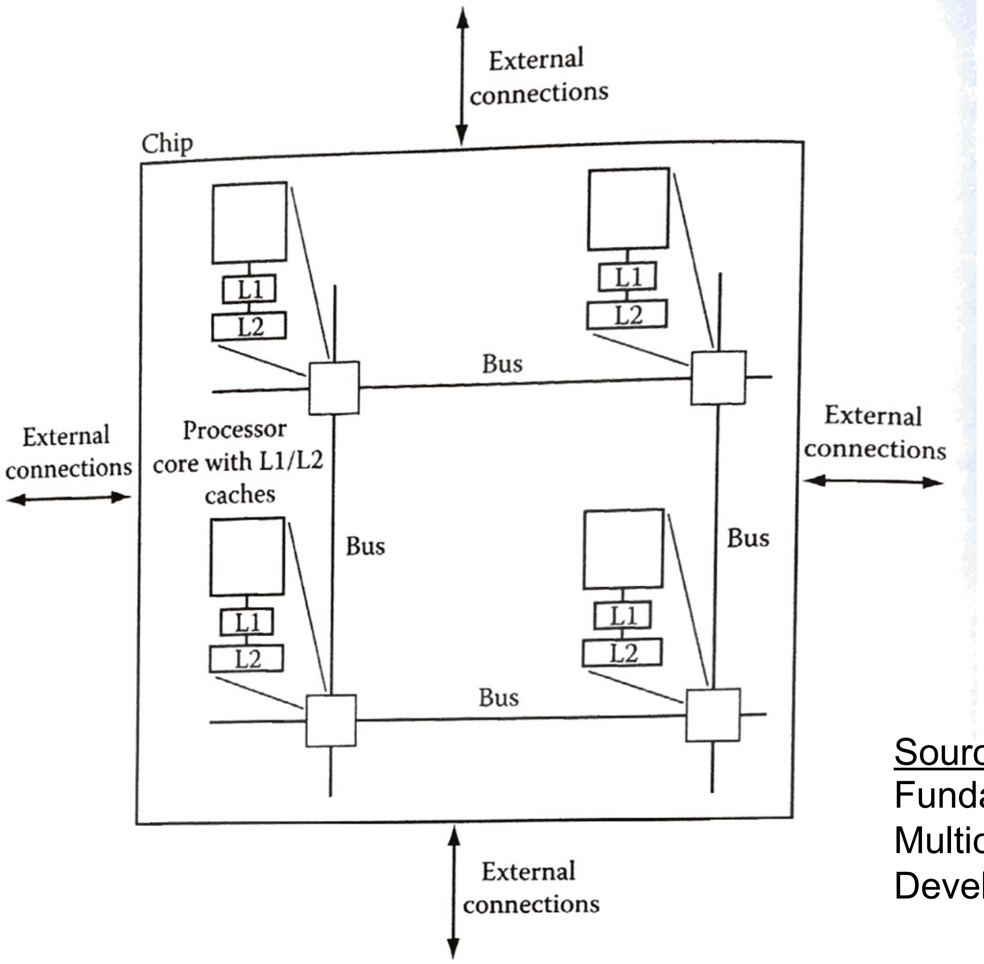
Thread-level Parallelism (high level)

Instruction-level Parallelism (low level)

Instruction-Level vs. Thread-Level Parallelism

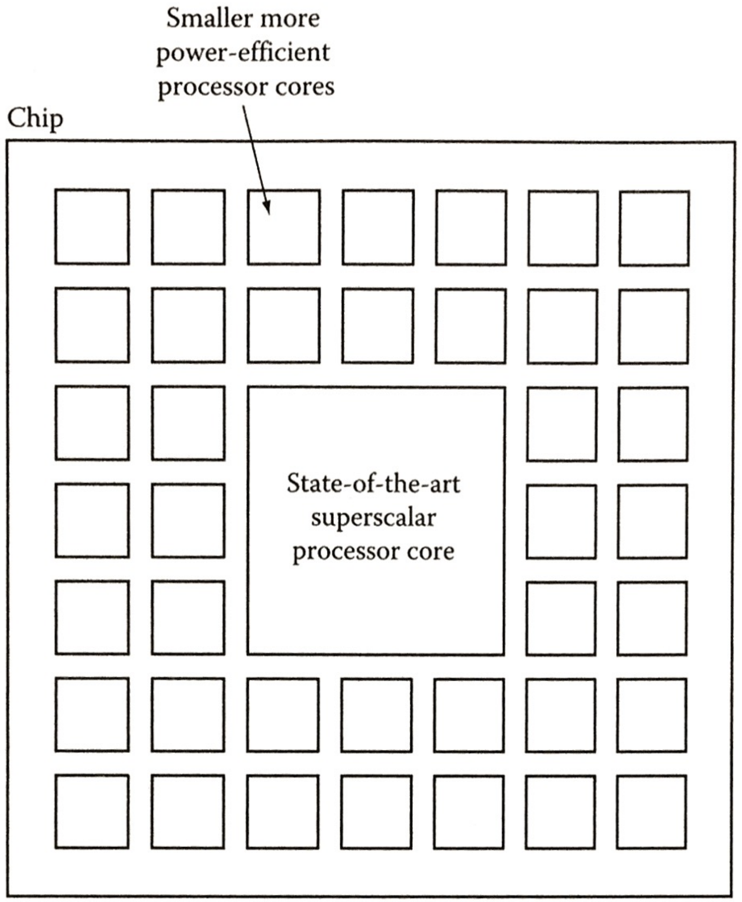
- Multithreading is an instruction-level approach to multi-threaded programs
  - Can be used on a single processor system
    - Switch between threads using fine-grained (between every instruction) or coarse-grained (during an expensive stall) multithreading
    - Need separate PC for each thread
    - Also need to separate memory, etc.
  - Hyperthreading is an Intel approach using Simultaneous multithreading (SMT)

Symmetric Multicore Design



<u>Source</u>:
Fundamentals of
Multicore Software
Development

Asymmetric Multicore Design



Source:
Fundamentals of Multicore Software Development

Massively Parallel Systems

- **GPU Computing**
  - 100s or 1000s of GPUs
- **Massively Parallel Processor Arrays (MPPAs)**
  - Array of 100s of CPUs + RAM
- **Grid Computing**
  - Nodes often perform different tasks
- **Cluster Computing**
  - Nodes often perform the same task

Introduction II

Summary

- Overview of multicore hardware

References

- “Computer Architecture: A Quantitative Approach” by Hennessy & Patterson
- “Fundamentals of Multicore Software Development” by Victor Pankratius & Ali-Reza Adl-Tabatabai & Walter Tichy

Next time

- Implicit Parallelism and OpenMP