ECSE 425 Lecture 26: Introduction to Multiprocessors

H&P Chapter 4

Last Time

- Multi-threading
  - Coarse-grained
  - Fine-grained
  - Simultaneous

Today

- Basic multi-processor architectures
- Parallel processing challenges
- Chapter 4.1-2

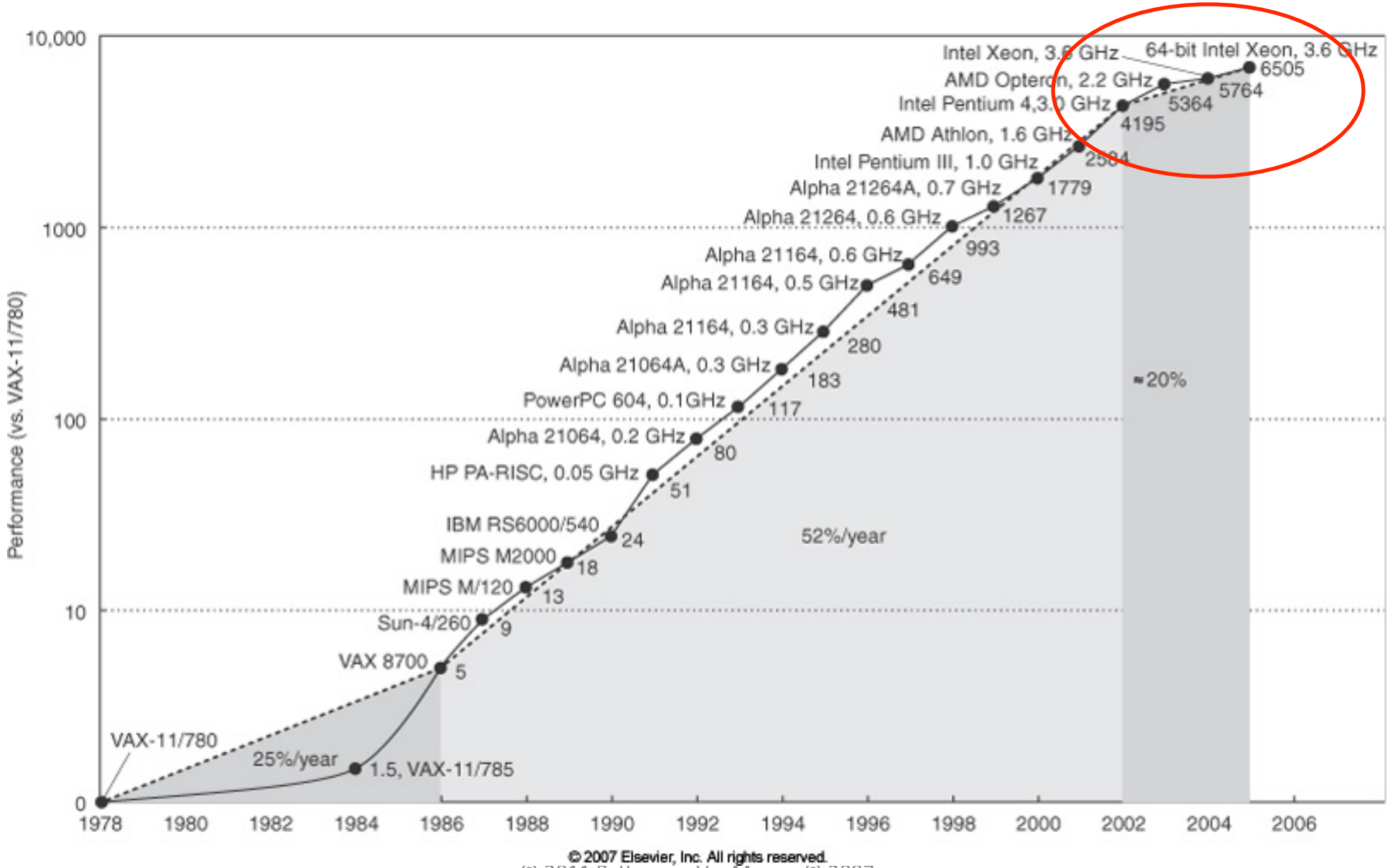
The Era of Multicore

"We are dedicating all of our future product development to multicore designs. We believe this is a key inflection point for the industry."

Intel President Paul Otellini

at Intel Developers Forum 2005

Uniprocessor Performance (SPECint)



Ushering in the Era of Multicore

- Growing interest in server performance
- Growth in data-intensive applications
- Increasing desktop performance is less important
- Greater understanding of effective multiprocessing
  - Especially in servers with significant natural TLP
- Advantage of leveraging previous design investment by replication
  - Rather than unique design

Flynn's Taxonomy

- In 1966, Flynn classified by data and control streams

Single Instruction Single Data **SISD** (Uni-processor)	**Single Instruction Multiple Data** **SIMD** (single PC: Vector, GPGPU)
Multiple Instruction Single Data **MISD** (No commercial machine so far)	**Multiple Instruction Multiple Data** **MIMD** (Clusters, SMP servers)

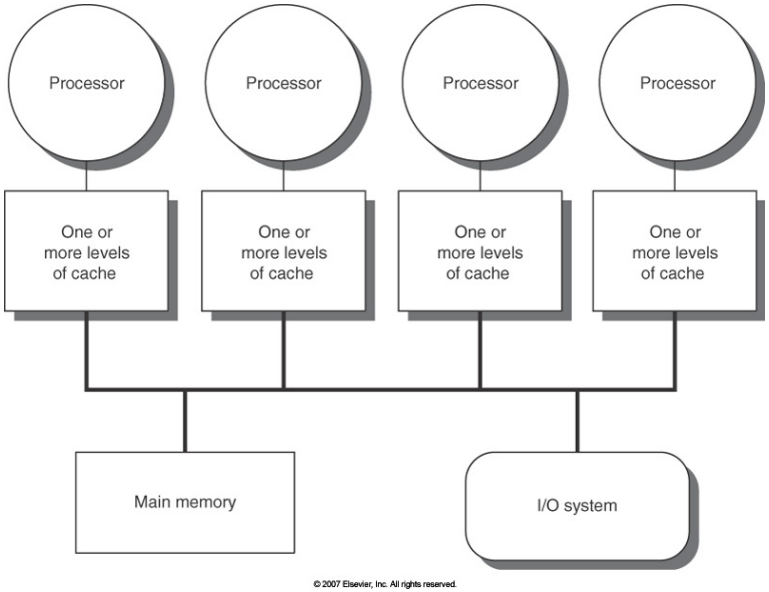
- SIMD ⇒ Data Level Parallelism
- MIMD ⇒ Thread Level Parallelism
- MIMD is popular because
  - Flexible: can run N programs or 1 multithreaded program
  - Cost-effective: same microprocessor in SISD & MIMD

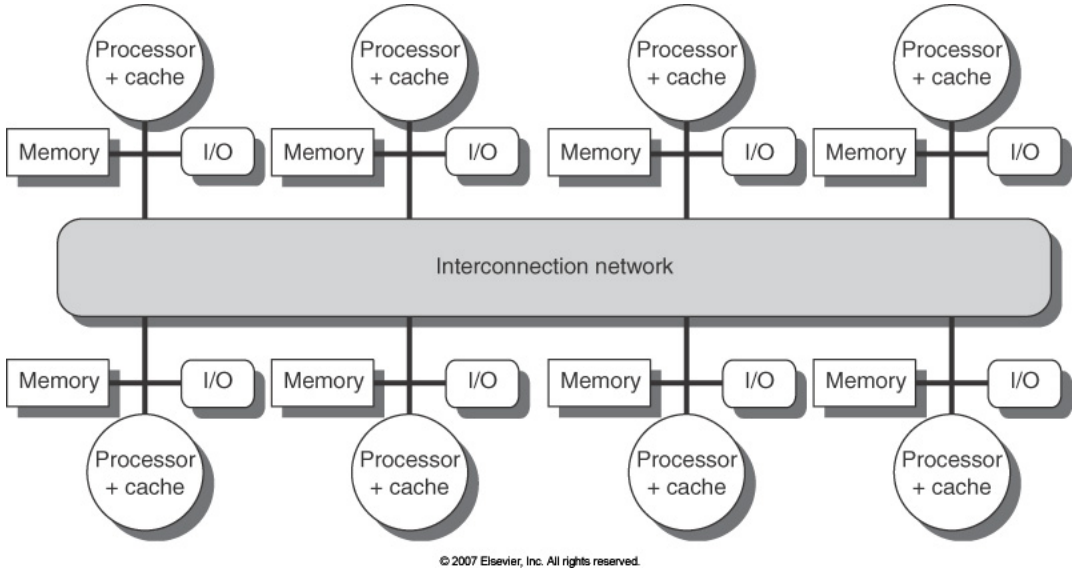
M.J. Flynn, "Very High-Speed Computers",
Proc. of the IEEE, V 54, 1900-1909, Dec. 1966.

Multiple processor computers

- "A parallel computer is a collection of processing elements that cooperate and communicate to solve large problems fast."
  - Parallel Architecture = Computer Architecture + Communication Architecture
- Two classes with different memory organizations
  - Centralized Memory Multiprocessor
    - < few dozen processor chips (and < 100 cores) in 2006
    - Small enough to share single, centralized memory
  - Physically Distributed-Memory multiprocessor
    - Larger number chips and cores than centralized
    - BW demands ⇒ Memory distributed among processors

Centralized vs. Distributed Memory





Centralized Memory

Distributed Memory

Scale

Centralized Memory Multiprocessor

- Symmetric Multiprocessors (SMPs)
- Uniform Memory Access (UMA)
  - All processors access the same memory location in the same amount of time
- Large caches ⇒ single memory can satisfy memory demands of small number of processors
- Can scale to a few dozen processors by
  - Using point-to-point connections or a switch
  - Adding many memory banks
- Not generally scalable: many bottlenecks!

Distributed Memory Multiprocessor

- Non-uniform Memory Access (NUMA)
  - Some memory addresses can be accessed more quickly than others; which, depends on the CPU
- Benefits:
  - Cost-effective way to scale memory bandwidth
    - If most accesses are to local memory
  - Reduces latency of local memory accesses
- Disadvantage:
  - More complex inter-processor communication

Memory Architecture

- Shared memory multiprocessors
  - Through a shared address space (via loads and stores)
    - Physical memory space can be separate
    - Same address on two processors refer to the same location
  - UMA for shared address, centralized memory MP
  - NUMA for shared address, distributed memory MP
- Message-passing multiprocessors
  - Explicit messages are passed among the processors
  - Inherent in clusters:
    - Same address on two processors refers to locations in two different memories

Parallel Processing Challenge One

- *The sequential fraction of execution*
  - We can only accelerate the parallel fraction
- Example: for 80x speedup from 100 cores, what fraction of the program can be sequential?
  - 10%
  - 5%
  - 1%
  - <1%

Sequential Portion: Amdahl's Law

$$\text{Speedup}_{\text{overall}} = \frac{1}{\left(1 - \text{Fraction}_{\text{parallel}}\right) + \frac{\text{Fraction}_{\text{parallel}}}{\text{Speedup}_{\text{parallel}}}}$$

$$\text{Fraction}_{\text{parallel}} = \frac{\left(1 - \frac{1}{\text{Speedup}_{\text{overall}}}\right)}{\left(1 - \frac{1}{\text{Speedup}_{\text{parallel}}}\right)}$$

$$\text{Fraction}_{\text{parallel}} = \left(1 - \frac{1}{80}\right)\left(1 - \frac{1}{100}\right)^{-1} = 99.75\%$$

Less than 1% of the program can be sequential!

Parallel Processing Challenge Two

- *The cost of communication*
  - Long access latency to remote memory
- Example:
  - 32 CPU MP, 2 GHz, 200 ns remote memory,
    - Remote access = 200/0.5 = 400 clock cycles.
  - All local accesses hit memory hierarchy
  - Base CPI is 0.5
- What is performance impact if 0.2% instructions involve remote access vs. none?
  - 1.5X
  - 2.0X
  - 2.5X

Addressing the Challenges

- Application parallelism
  - Solved with user software (not compilers)
  - Parallel programming platforms (*e.g.*, OpenMP)
  - Parallel versions of algorithms
- Long remote latency impact
  - Solved by both the architect and programmer
  - Reduce frequency of remote accesses either by
    - Caching shared data (HW)
    - Restructuring the data layout to make more accesses local (SW)
  - Also tolerate latency by multithreading or prefetching
- We will focus on reducing the impact of remote communication latency

Summary

- Basic multi-processor architectures
  - Centralized
  - Distributed
- Parallel processing challenges
  - The sequential fraction of software
  - The latency of remote memory accesses

Next Time

- Cache coherency and memory consistency
  - Read Chapter 4.2!