Multiprocessors, Multiprocessing, and Multithreading

more is better?

Multiprocessors

• why would you want a multiprocessor?
• what things can it do well?
• What things can’t it do well?
• What things can it do that a bunch of computers can’t do?
• How much are you willing to pay?

Classifying Multiprocessors

• Interconnection Network
• Memory Topology
• Programming Model

Interconnection Network

• Bus
• Network
• pros/cons?
Memory Topology

- UMA (Uniform Memory Access)
- NUMA (Non-uniform Memory Access)
- pros/cons?

Programming Model

- Shared Memory -- every processor can name every address location
- Message Passing -- each processor can name only it's local memory. Communication is through explicit messages.
- pros/cons?

- find the max of 100,000 integers on 10 processors.

Parallel Programming

- Shared-memory programming requires synchronization to provide mutual exclusion and prevent race conditions
  - locks (semaphores)
  - barriers

Multiprocessor Caches (Shared Memory)

- the problem -- cache coherency
- the solution?
Cache Coherency

- **write-update**
  - on each write, each cache holding that location updates its value
- **write-invalidate <= most common**
  - on each write, each cache holding that location invalidates the cache line.

- both schemes MUCH easier on a bus-based multiprocessor
- potentially requires a LOT of messages, but...

A good cache coherency protocol can avoid sending unnecessary (and expensive) invalidate or update messages.

Allows each cache line to be in one of several states.

- MESI (Illinois)
  - modified
  - exclusive
  - shared
  - Invalid

What happens:
- Load miss?
- Store hit to exclusive line? Modified? Shared?

How do you know when an external read/write occurs?

- Snooping protocols
- Directory protocols

(1996 Thesis Slides)
Hardware Multithreading

Conventional Processor

Multithreaded

**Superscalar Execution**

Issue Slots

Vertical waste

Horizontal waste

**Superscalar Execution with Fine-Grain Multithreading**

Issue Slots

Thread 1

Thread 2

Thread 3

**Simultaneous Multithreading**

Issue Slots

Thread 1

Thread 2

Thread 3

Thread 4

Thread 5
**SMT Performance**

- **Simultaneous Multithreading**
- **Fine-Grain Multithreading**
- **Conventional Superscalar**

**Multithreaded Processors**

- Coarse-grain multithreading (Alewife-MIT)
  - context switch at long-latency operations (cache misses)
- Fine-grain multithreading (Tera Supercomputer)
  - context switch every cycle
- Simultaneous multithreading (SMT) (Tullsen, Eggers, Levy 1995)
  - execute instructions from multiple threads in the same cycle
  - is only different from fine-grain multithreading in the context of superscalar execution
  - requires surprisingly few changes to a conventional out-of-order superscalar processor
  - Was announced to be featured in the next Compaq Alpha processor (21464), but that processor never completed.
  - Introduced in the Intel Pentium 4 processor – announced as “Hyper-threading technology.” (HT Technology)
  - IBM Power 5, 6 has 2 cores, each 2-way SMT.
- Sun Niagara not SMT, but has eight cores, each capable of running four threads.

**Multi-Core Processors**

- Multiple cores on the same die, may or may not share L2 cache.
- Intel, AMD, IBM all currently have dual-core processors (some SMT as well). Intel, AMD just announced quad-core. Sun Niagara 4x8.
- Everyone’s roadmap seems to be increasingly multi-core.

**Multiprocessors -- Key Points**

- Network vs. Bus
- Message-passing vs. Shared Memory
- Shared Memory is more intuitive, but creates problems for both the programmer (memory consistency, requiring synchronization) and the architect (cache coherency).
- Multithreading gives the illusion of multiprocessing (including, in many cases, the performance) with very little additional hardware.
- When multiprocessing happens within a single die/processor, we call that a chip multiprocessor, or a multi-core architecture.