2008
YEAR
PRESENTATION
The University of Auckland | New Zealand

Computer Science 703

Advance Computer Architecture

2006 Semester 1

Lecture Notes 5

18Mar08

Cache Coherence; Speculation

James Goodman

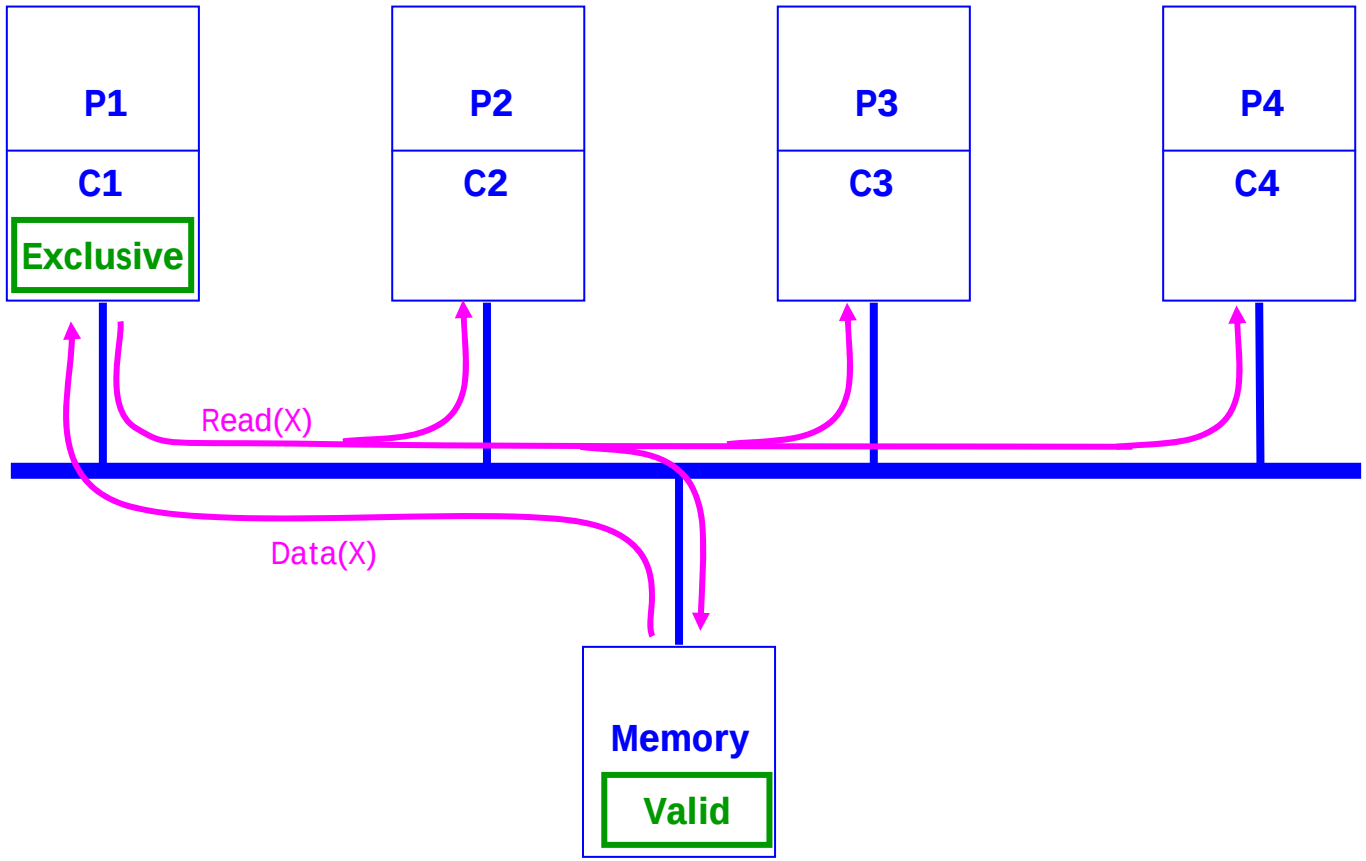
Department
of
Computer Science

Improvements

- Read to Uncached
  - Go from I to E (not S)
- Advantage: Can write without invalidation

Cache Coherence

Read to Uncached



Summary of Snooping Caches

- Snooping cache coherence protocols are the *dominant multiprocessor technique* used today
- Most microprocessors conform to snooping cache protocols (e.g., Intel Pentium: up to 4 processors on the bus)
- Snooping has been extended to much larger systems by a series of creative methods, but scalability is fundamentally limited by broadcast requirements

Coherency in Multiple-Bus Systems

- Scalable protocols involve maintaining a directory auxiliary information keeping track of which caches have copies of which cache lines
- Directory-based scheme can use point-to-point connections, which potentially have both higher speed and much higher bandwidth than a bus
- Because of the additional delay (typically three hops for most transactions), only very large systems benefit from directory-based schemes.
- There have been several commercial products, but so far, no directory-based scheme has been highly successful.

Terminology

- Snooping-based schemes have been extended beyond a single bus (maintaining the notion of a single "logical" bus). Broadcast-based schemes are called *Symmetric MultiProcessing* (SMP)
- Directory-based schemes continue to be widely studied, and there are many variations proposed and some products. Such schemes are often referred to as *Non-Uniform Memory Access* (NUMA) systems.

Speculation

Motivation for Speculation

- For performance, instructions are overlapped
- Some instructions may finish before earlier instructions have finished.
  - What happens if an earlier instruction causes a trap?
  - IBM System/360: the "imprecise interrupt": abort and let the programmer deal with the problem
- Virtual memory causes traps that cannot cause the program to abort

Elimination of Imprecise Interrupt

- Issue instructions out of order
- They may "complete" out of order
- They "commit" in-order (or simultaneously)

Why Speculate?

- Instructions have *dependences*
  - Data dependence: instruction can't issue because operand is not ready
  - Control dependence: not sure which instruction to issue next
- Both often result in cache miss

Control Dependence: Branch Prediction

- If we can accurately predict which way a branch will go, we can begin execution (or at least, fetch next instruction) earlier
- Inconvenient truth: we can't predict branches with high accuracy

The Cost of Speculation

- Prefetching instructions (in a pipeline) is speculation!
  - When a branch occurs, the following instruction will not be executed – we speculatively fetched it.
  - Costs
    - Fetching non-executed instructions may prevent fetching of executed instructions
    - Fetched instructions not executed need to be replaced by executable instructions
  - Goal: prefetch instructions when the expected outcome saves time, i.e., when

$$E(\text{time\_saved}) \times P(\text{success}) > E(\text{time\_lost}) \times P(\text{failure})$$

How to predict branch decision?

- Brute force: fetch down both paths
- Statically
  - Branch type
    - Special instructions for loop variables
    - Software may predict
  - Forward not take, backward taken
- Dynamically
  - What happened last time(s)?
  - How did we get here?