

Shared Memory Consistency Models: A Tutorial

Sarita V. Adve, Kourosh Gharachorloo

Jamey Sharp

2008/02/18

Outline

- 1 Overview
- 2 Optimizations
- 3 Models

Outline

1 Overview

2 Optimizations

3 Models

Memory consistency models

- The input to memory is a set of memory operations (reads and writes) partially ordered by program order.
- The output of memory is the collection of values returned by all read operations.
- A consistency model is a function that maps each input to a set of allowable outputs. . . .
- The program must be written to work correctly for any output allowed by the consistency model.

“A Unified Theory of Shared Memory Consistency”
Steinke and Nutt

Hardware

CPUs are fast. Memory is slow. So:

- Caches
- Store buffers
- Out-of-order instruction issue

Outline

1 Overview

2 Optimizations

3 Models

Store-buffer forwarding

- Idea: Do more computation while waiting for store to complete
- Store-buffer is a queue of unretired stores
- Why not satisfy a local read out of the store buffer?
- Just like cache but without any coherency protocol at all. . .

Non-blocking read

- Idea: Do more computation while waiting for load to complete
- Might need to do another load
- Later load might hit local cache while earlier load stalls waiting for main memory

Overlapping writes

- Different regions of memory may have different memory controller queues (NUMA?)
- One region might be idle while another is busy
- Boom, recipe for store reordering

Outline

1 Overview

2 Optimizations

3 Models

Sequential Consistency

- “*Sequential*”: “The result of an execution is the same as if the operations had been executed in the order specified by the program.”
- “*Sequentially consistent*”: “The result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by the program.”

“How to Make a Multiprocessor Computer That Correctly
Executes Multiprocess Programs”

Leslie Lamport

Classification

- Relax Write-to-Read program order
- Relax Write-to-Write program order
- Relax Read-to-Read and Read-to-Write program orders
- Read others' writes early
- Read own writes early

x86

- 1 Loads are not reordered with other loads.
- 2 Stores are not reordered with other stores.
- 3 Stores are not reordered with older loads.
- 4 Loads may be reordered with older stores to different locations.
- 5 Memory ordering obeys causality (AKA transitive visibility).
- 6 Stores to the same location have a total order.
- 7 Locked instructions have a total order.
- 8 Loads and stores are not reordered with locked instructions.
- 9 Stores may become locally visible before they are remotely visible.

“Intel 64 Architecture Memory Ordering White Paper”

Summary

- Multi-threaded program correctness depends on the memory consistency model of the architecture.
- Sequential consistency is too slow.