SHARED MEMORY SYSTEMS

Mahdi Nazm Bojnordi

Assistant Professor
School of Computing
University of Utah
Overview

- Shared memory systems
  - Inconsistent vs. consistent data
- Cache coherence with write back policy
  - MSI protocol
  - MESI protocol
- Memory consistency
  - Sequential consistency
Recall: Shared Memory Systems

- Multiple threads employ a shared memory system
  - Easy for programmers
- Complex synchronization mechanisms are required
  - Cache coherence
    - All the processors see the same data for a particular memory address as they should have if there were no caches in the system
    - e.g., snoopy protocol with write-through, write no-allocate
      - Inefficient
  - Memory consistency
    - All memory instructions appear to execute in the program order
    - e.g., sequential consistency
Snooping with Writeback Policy

**Problem:** writes are not propagated to memory until eviction
- Cache data maybe different from main memory

**Solution:** identify the owner of the most recently updated replica
- Every data may have only one owner at any time
- Only the owner can update the replica
- Multiple readers can share the data
  - No one can write without gaining ownership first
Modified-Shared-Invalid Protocol

- Every cache block transitions among three states
  - Invalid: no replica in the cache
  - Shared: a read-only copy in the cache
    - Multiple units may have the same copy
  - Modified: a writable copy of the data in the cache
    - The replica has been updated
    - The cache has the only valid copy of the data block

- Processor actions
  - Load, store, evict

- Bus messages
  - BusRd, BusRdX, BusInv, BusWB, BusReply
MSI Example

![Diagram showing MSI example with states invalid to shared and transitions between them labeled Load/BusRd.](image-url)
MSI Example

Invalid \(\rightarrow\) Shared

Load/BusRd

BusRd/[BusReply]

Load/--
MSI Example

- **invalid** to **shared** via **Load/BusRd**
- **shared** to **invalid** via **Evict/--**
- **shared** to **BusRd/[BusReply]**
- **Load/--**

**Diagram:**
- Two processes (P1, P2) connected to a bus (BUS) with an evict action.
MSI Example

- **Load/BusRd**
- **BusRd/[BusReply]**
- **Evict/**
- **Load/**

States:
- **invalid**
- **shared**
- **modified**

Actions:
- Load, Store/--
- Store/BusRdX
- BusRdX/[BusReply]
MSI Example

Load, Store/---

Load/BusRd

BusRdX/[BusReply]

Evict/---

Load/---

invalid

shared

modified

Load

P1

I

P2

M

BUS
MSI Example

Graph showing transitions between states:
- Invalid
- Shared
- Modified

Transitions:
- Load/BusRd to Invalid
- BusInv, BusRdX/[BusReply] to Invalid
- Evict/-- to Shared
- Load/-- to Shared
- BusRd/[BusReply] to Shared
- Store/BusRdX to Modified
- BusRd/BusReply to Shared
- Store/BusInv to Modified
- Load, Store/-- to Invalid

Nodes:
- P1
- P2
- S
- BUS

States:
- Invalid
- Shared
- Modified
MSI Example

Ivalid

BusInv, BusRdX/[BusReply]

BusRd/[BusReply]

Load/BusRd

shared

Evict/--

BusRd/BusReply

Load/--

modified

BusRdX/BusReply

Store/BusRdX

invalid

Store/BusInv

Load, Store/--

P1

P2

M

I

BUS

Load/

Store/
MSI Example

Invalid state:
- Load/BusRd
- BusInv, BusRdX/[BusReply]

Shared state:
- Load/--
- BusRd/[BusReply]

Modified state:
- Store/BusRdX
- BusRdX/BusReply
- BusRd/BusReply
- Store/BusInv

Transition arrows indicate the actions and states in the MSI protocol:
- Load, Store/--
- Evict/--
Modified, Exclusive, Shared, Invalid

- Also known as Illinois protocol
  - Employed by real processors
  - A cache may have an exclusive copy of the data
  - The exclusive copy may be copied between caches

- Pros
  - No invalidation traffic on write-hits in the E state
  - Lower overheads in sequential applications

- Cons
  - More complex protocol
  - Longer memory latency due to the protocol