Coherence, Locking and Consistency – Part II

Dr. A. Sahu
CSE, IIT Guwahati

Multicore Difficulties I

• Multiprocessors are likely to be cost/power effective solutions
  – Because it share lots of resources
    • Personal room is costlier than dormitory
  – Sharing resource arise many other problems
    • Coherence
      – Shared data at all placed should be same
    • Critical Sections
      – Lock and Barrier Design
    • Consistency
      – Order should be similar to serial (ROB)
    • One processor Interference others
      – Share efficiently using some policy

Which procs/caches communicate?

• Snoopy protocol
  – Broadcast invalidate or update messages
  – All processors snoop on the bus
• Directory based protocol
  – Maintain directory - list of copies
  – Communicate selectively
  – Directory - centralized (memory) or distributed (caches)

Status of each cache block?

valid/invalid  private/shared  clean/dirty

• Simplest protocol (3 states)
  Invalid, (shared) clean, private dirty
• Berkeley protocol (4 states)
  Invalid, (shared) clean, private dirty, shared dirty
• Illinois protocols (4 states)
  Invalid (I), shared clean(S), private clean (E), private dirty (M) : MESI
• MOESI : Owner responsible for copy back to memory

Simplest invalidation protocol

Use 3 states : Invalid, shared clean, private dirty

Are 3 states sufficient? No

• All writes in clean state cause invalidation traffic (whether shared or not)
  Introduce private clean state (e.g. Illinois protocol) - reduce invalidation traffic
• Clean copies are always supplied by memory
  Introduce shared dirty state (e.g. Berkeley protocol) - permit copies to be supplied by caches which own the required blocks
  Illinois protocol also permits a cache to supply the copy, but if it is coming from a dirty state, memory is also updated

Part of ACA Course @IITG
4-state Berkeley protocol

Use 4 states: Invalid, clean, private dirty, shared dirty

4-state Illinois protocol: MESI

Private dirty (Modified), Exclusive clean, Shared clean and Invalid

Read Miss in Illinois Protocol

Same as 3-state protocol

Write Hit in Illinois Protocol

If some other cache has a copy

Write Miss in Illinois Protocol

no other cache has a copy

preceded by writing back a block if previous state was "dirty (M)"
**Write Miss in Illinois Protocol**

- **P**: Present
- **M**: Modified
- **C**: Clean
- **I**: Invalid

- Any **M** cache block is replaced by writing back a block if the previous state was "dirty" (M).
- Data supplied by the owner is written back as "dirty" (M) in the **I** cache.
- Also, write to memory if the previous state was "dirty" (M).

**MOESI : Improved MESI**

- **Exclusive**:
  - Hold the most recent, correct copy of the data.
  - The data in main memory is also the most recent, correct copy of the data.
  - No other processor holds a copy of the data.
  - The cache line may be changed to the Modified state at any time in order to modify the data.
- **Shared**:
  - Holds the most recent, correct copy of the data. Other processors in the system may hold copies of the data in the Shared state.
  - Line may be dirty with respect to memory;
  - If it is, some cache has a copy in the Owned state, and that cache is responsible for eventually updating main memory.
- **Invalid**

**Directory Based Coherence Protocols**

- Suitable for distributed shared memory
- No broadcasting and snooping (common bus not required)
- Communicate selectively using a directory
- Directory maintains lists of copies of each block
- Directory location - centralized (memory) or distributed (caches)

**Directory Organizations**

**Structure**
- Central Directory (CD)
  - Memory is updated
  - May be physically distributed
- Distributed Directory (DD)
  - Memory is not updated
  - DD-INV:
    - SDD (Scalable Dist. Directory)
    - SCI (Scalable Coherent Interface Protocol) – proposed IEEE std

**Actions for caches**
- Invalidate
- Update

**Centralized Directory Structure**

- **Directory**
  - One bit vector per block
  - cache 0
  - cache 1
  - cache 2
  - cache 3

A Sahu
Distributed Directory Structure

Directory

one pointer per cache & mem block

Singly Linked List (SCI has doubly linked list)

Centralized Directory Protocol

Directory

miss reply w/ INV count

miss

reply

WR

I-ACK

I-ACK

I-ACK

I-ACK

Singly Linked List (SCI has doubly linked list)

Centralized Directory Protocol in a MESH

Directory

miss reply w/ INV count

miss

reply

WR

I-ACK

I-ACK

I-ACK

I-ACK

Singly Linked List (SCI has doubly linked list)

Synchronization

• Processes run on different processors independently
• At some point they need to know the status of each other for
  – Communication, mutual exclusion etc
• Simple Locking

Lock (L)
Critical Section (C)
Unlock (L)

Hennessy Book 5th Ed. Sec 5.5, Culier Sec 5.5
Synchronization

- Simple Locking
  ```
  lock:  ld reg, loc   // copy location to reg
cmp reg #0   // compare with 0
bnz lock    // if not zero try again
st loc #1   // store 1 at loc to mark it locked
return;
  ```
- Unlock:  st loc #0

Synchronization Problem

Hennessy Book 5th Ed. Sec 5.5. Culler Sec 5.5
- Hardware primitive for atomic read+write is required e.g.
  - Test & Set,  // test for unlock (0) then set the lock (1)
  - Exchange,
  - Fetch & Increment

Spin Lock with Exchange Instr.

- Lock:  0 indicates free and 1 indicates locked
- Code to lock X:
  ```
  r2 ← 1
lockit: r2 ↔ X    ; atomic exchange
if(r2#0) → lockit  ; already locked
  ```
- Locks are cached for efficiency, coherence is used // to take care of timing
- Better code to lock X:
  ```
  lockit: r2 ↔ X    ; read lock
  if(r2#0) → lockit ; not available
  r2 ← 1
  r2 ↔ X    ; atomic exchange
  if(r2#0) → lockit ; already locked
  ```

LD Locked & ST conditional

- LL r1 X  Reading from a location X
  //do some operation
- SC r3 X  Storing to location r3 to X
  if unsuccessful r3 == 0
- Store will be unsuccessful if values of X is altered/changed by others processor between time of LL and time of SC
- Load linked/Store Conditional

Imagine lock is Mem Address, How bad performance will be