An Adaptive Cache Coherence Protocol Optimized for Migratory Sharing

Per Stenström, Mats Brorsson, and Lars Sandberg

Presented by Allen Lee

April 2, 2008

Observations

- Data structures accessed within critical sections are read and modified by exactly one processor at any given time.
  - Also called *migratory objects* [Gupta and Weber].
- Each processor that accesses these data structures typically causes a cache miss followed by an invalidation request.
  - These transactions can be combined.

Migratory Sharing

- R_i denotes a read access by processor *i*
- W_i denotes a write access by processor *i*
- Consider the access pattern, where i != j,

$$(R_i)^+(W_i)(R_i \mid W_i)^*(R_j)^+(W_j)(R_j \mid W_j)^* \ldots$$

- A block that adheres to the above pattern is called *migratory*, otherwise it is *ordinary*.

Detecting Migratory Sharing

- A block is nominated as migratory if
  1. A *read-exclusive* (invalidation) request comes a processor different from the one that most recently wrote to it, and
  2. There are exactly two copies of the block.

$$(R_i)^+(W_i)(R_i \mid W_i)^*(R_j)^+(W_j)(R_j \mid W_j)^* \ldots$$

Condition 1

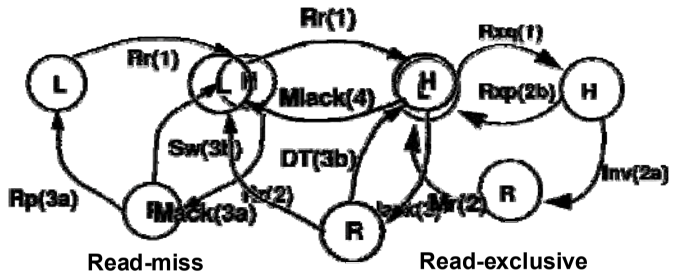
Condition 2

Review: Stanford DASH

- Directory-based write-invalidate protocol
- Home keeps track of global state of blocks
  - *Uncached*
  - *Shared-Remote*
  - *Dirty-Remote*
- A block in a local node is either
  - *Invalid*
  - *Shared*
  - *Dirty*

Review: Stanford DASH

(L = local, H = home, R = remote)



Rr = Read miss

Rp = Response from remote to local

Sw = Response from remote to home

Rxq = Read-exclusive request

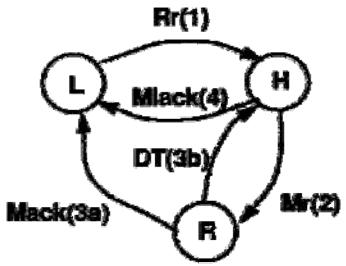
Rxp = Read-exclusive response

Inv = Invalidate

Iack = Invalidation acknowledgment

Actions for Migratory Blocks

(L = local, H = home, R = remote)



Rr = Read miss

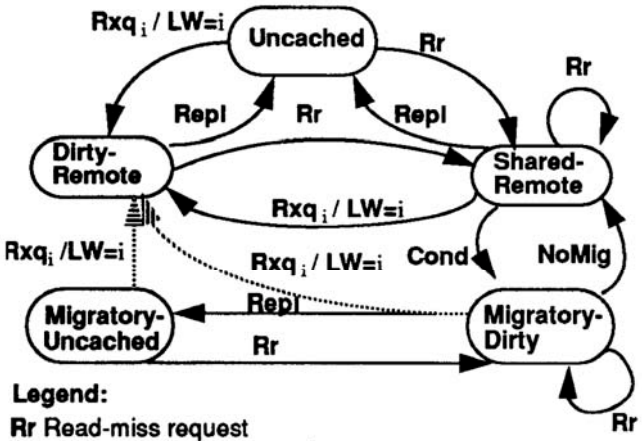
Mr = Migratory read-miss request

Mack = Remote gives up ownership and sends copy to local (migratory ack)

DT = Notification of ownership transfer

MIAck = Home has updated directory

Detecting Migratory Blocks



Legend:

Rr Read-miss request
Rxq Read-exclusive request
Repl Replacement request
Cond Rxq i && N==2 && LW != i
LW Last-writer pointer

Migratory to Ordinary

- Consider the following access pattern

  $Rr_i, Rr_j, Rr_i, Rr_j, \ldots$
- To avoid ping-pong, introduce *Migrating* local cache state
  - Initial read causes local state to be *Migrating* instead of *Dirty*
  - Subsequent write changes state to *Dirty*
  - Receiving migrating read request (Mr) in *Migrating* generates NoMig signal and the block is thereafter considered ordinary

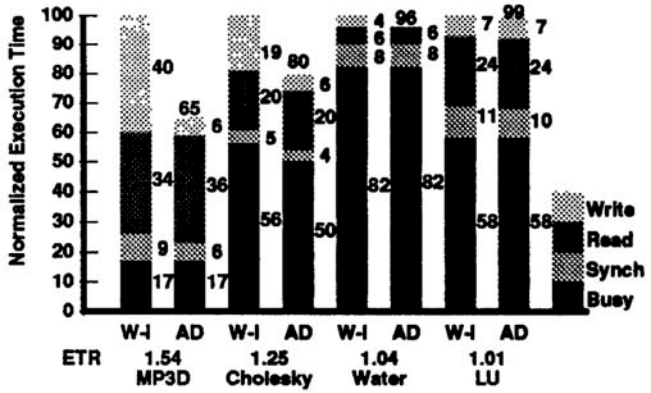
Architectural Model

- 16 nodes interconnected by two 4x4 wormhole-routed meshes
- 64-Kbyte cache with 16-byte line size
- 100 MHz processor clock

Latency numbers (1 pclock = 10 ns).

Latency for Read and Write Requests	Time
Hit in Cache	1 pclock
Fill from Local Memory	22 pclocks
Fill from Remote (2-hop)	54 pclocks
Fill from Remote (3-hop)	73 pclocks
Read-Exclusive Request to Remote (2-hop)	51 pclocks
Read-Exclusive Request to Remote (3-hop)	70 pclocks

Results



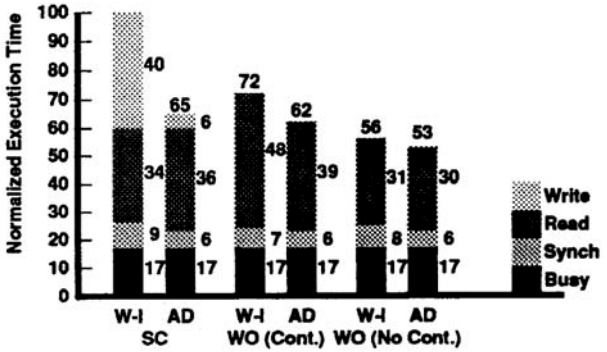
Application	Read-excl. Req. Reduction
MP3D	87%
Cholesky	69%
Water	96%
LU	5%

Results – Traffic Reduction

- 704 bits are sent under W-I to serve read-miss and subsequent read-exclusive request
- 328 bits for the adaptive protocol

Application	Traffic Reduction
MP3D	32%
Cholesky	22%
Water	31%
LU	1%

Results – Effect on Consistency Model



- Relaxed consistency models require high bandwidth for global requests.
  - Traffic reduction results in better performance

Results – Impact of Cache Size

- Smaller caches have higher replacement miss rates.
- Under AD, smaller cache sizes cause the write penalty reduction to be smaller.

Table 4: Write penalty reduction (WPR) of W-I by AD and repl. miss-rates (MR) for 64 and 4Kbyte caches.

	MP3D	Cholesky	Water	LU
64 Kbyte — MR	3%	3%	3%	3%
4 Kbyte — MR	7%	18%	9%	21%
64 Kbyte — WPR	86%	67%	94%	3.7%
4 Kbyte — WPR	67%	32%	85%	0.2%

Conclusion

- The adaptive protocol optimizes coherence actions due to migratory sharing
- Technique requires a simple extension of directory-based write-invalidate protocol
- Improves performance by reducing write stall time and overall network traffic