

Cashmere-2L

Peter Andrews and Justin Steinberg

Goals

- Take advantage of built-in hardware coherence whenever possible
- Use software coherence when necessary
- Overcome the problem of high costs of communication
- Highly asynchronous

The Idea

- Two-level coherence protocol
 - Use a combination of hardware and software coherence
- “Moderately” Lazy Release Consistency
 - Requires Data Race Free (DRF) code
 - Invalidations take effect at the time of the next acquire
- Directory-based coherence protocol
- Exclusive mode
- Timestamps to maintain temporal ordering
 - Used for optimizations
- In and out diffs

The Hardware

- Each node is 1 AlphaServer machine consisting of 4 processors
- 8 nodes connected by Memory Channel (MC)

Memory Channel (MC)

- Low latency remote writes
- No remote reads
- 32-bit granularity
- Transmit regions – mapped to I/O
- Receive regions – mapped to RAM
- Guaranteed write ordering

Implementation Details

Write Notices

- *Used to notify other nodes of page modifications*
- Multi-bin, two-level structure
- Each node has a global list with 7 receive bins
- Each processor has its own (lock-protected) write notice list
- When appropriate (acquires), processors move write notices from the global list to the processors' local lists
- Bit-map to avoid redundancy

Acquires

- Recall: Invalidations take effect at the time of the next acquire
- Process write notices
 - Processor goes through write notices in the global bins and distributes them to the local processors' private lists
 - Update timestamp of latest write notice
- If a write notice for the current page exists with a timestamp more recent than the latest local update, the page must be invalidated

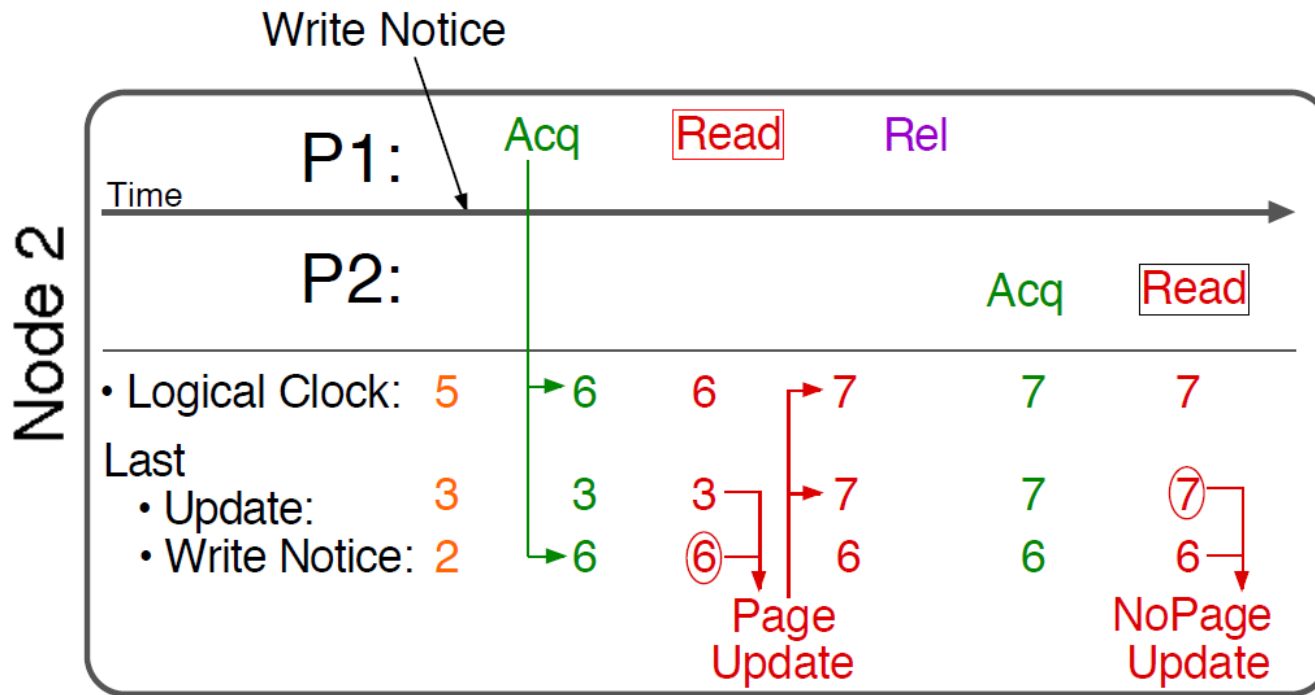
Releases

- Flush all dirty, non-exclusive pages to the home node
- Send write notices for each dirty page to all nodes that aren't the home node
- Optimization: If two processors on a same node are releasing at the same time, only one of them needs to flush changes for a given page
 - This is accomplished by comparing timestamps

Page Faults

- *Entry point to software coherence*
- Fetch the page from the home node when necessary
- If page is being shared, add to dirty list
- Otherwise, put the page in exclusive mode

Optimization Example



Dirty Page List

- Private to each processor
- Contains a list of pages that are known to have been modified since the last release operation
- Used during a page flush
 - Out diff

Exclusive Mode

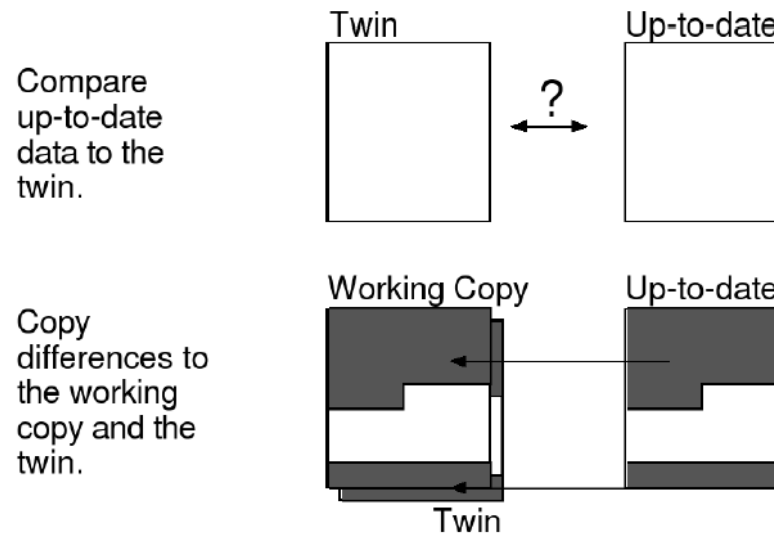
- *No* extra overhead for maintaining coherence
- When another node requests a page in exclusive mode, the faulting processor requests a page flush
- The holder also removes the page from exclusive mode and supplies a copy of the page to the faulting processor
- Future requests for the page are handled by the home node, as per usual

Home node

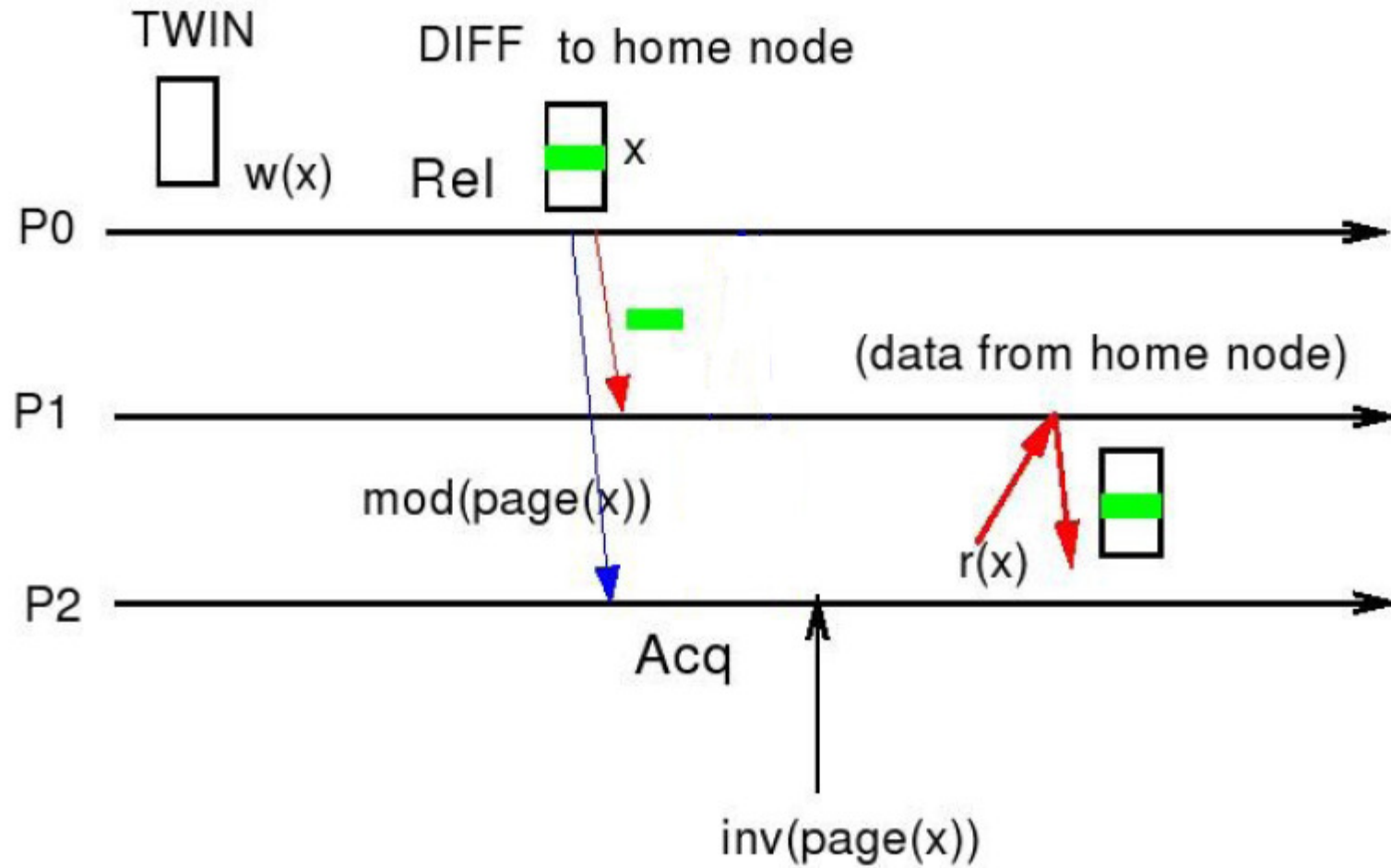
- The origin of a particular page
- Initially assigned fairly randomly in a round-robin manner
- Re-assigned on first use to the first node that uses it
- Note: It requires a lock to relocate the page, but this is only done once per page.

Twins and Diffs

- Represent the local node's view of the master copy on the home node
- *Outgoing diff* used on the *page* and its *twin*
- *Incoming diff* used on the *twin* and the *fresh copy*



Example



Global Directory

- Two-level structure
- Each page's entry is 8 32-bit words
 - 1 entry per node
- Contains page permissions, exclusive mode information, and ID of home node
- There is some redundant information
 - Why?

Second-level Directory

- Contains page information (Invalid? Read only? Read/write?)
- Contains 3 (logical) timestamps
 - Last home-node flush
 - Last local update
 - Last received write notice

Locks

- 8-entry arrays (sounds familiar?)
- Acquire
 - A node must gain access to the array via a bit flag using Load-linked/Store-Conditional (ll/sc)
 - Then, the node should write its own entry and request a copy of the entire array.
 - If other nodes' regions are set, clear own entry, back off, and try again.
- Release
 - A node simply clears its own entry.

Barriers

- Two-level structure
- Within each node, each processor flushes pages for which it is the last arriving local writer
- When an entire node has arrived, it writes to its entry in a similar data structure as a Lock (8-entry array)

Results

- Note the 2L version has more costly synchronization operations than the 1L version

Operation	Time (μs)	
	2L/2LS	1LD/1L
Lock Acquire	19	11
Barrier	58 (321)	41 (364)
Page Transfer (Local)	-	467
Page Transfer (Remote)	824	777

- Why is this an improvement over the 1L version?
- “In comparison to a one-level protocol that does not share memory in hardware within an SMP, Cashmere-2L improves performance by up to 46%.”

More Results

- Execution time improved by 5% using the largely lock-free data structures when compared to the compressed, lock-based structures.

Summary

- Two-level structure
 - Improvement over Cashmere-1L
 - Treated each processor as a separate node
 - Hardware and Software coherence
- Low latency writes
- Home-based protocol
- Moderately Lazy Release Consistency
- Multiple writers
- Two-way diffing

Cashmere vs. TreadMarks

Similarities

- Shared memory coherence protocol
- No kernel modifications required (although Cashmere does some optimizations)
- Both utilize twins and diffs
- Both require DRF code

Differences

- Release consistency
- Type of timestamps
- How to obtain a page
- TreadMarks uses a one-level protocol
- Home node

Spelling lesson

- Consistancy \neq Consistency