

Portland State University
ECE 588/688

Directory-Based Cache Coherence Protocols

Why Directory Protocols?

- Snooping-based protocols may not scale
 - ◆ All requests must be broadcast to all processors
 - ◆ All processors should monitor all requests on the shared interconnect
 - ◆ Shared interconnect utilization can be high, leading to very long wait times
- Directory protocols
 - ◆ Coherence state maintained in a directory associated with memory
 - ◆ Requests to a memory block do not need broadcasts
 - Served by local nodes if possible
 - Otherwise, sent to owning node
- Note: Some snooping-based protocols do not require broadcast, and therefore are more scalable

Design Issues for Distributed Coherence Protocols

- Correctness
 - ◆ Memory consistency model: Performance vs. Ease of programming
 - ◆ Deadlock avoidance
 - ◆ Error handling (fault tolerance)
- Performance
 - ◆ Latency
 - ◆ Bandwidth
- Distributed Control and Complexity
- Scalability

The Stanford DASH Prototype

- Directory Architecture for SHared memory
- Architecture consists of many clusters
 - ◆ Each cluster contains 4 processors
 - ◆ Processor caches
 - L1I: 64KB, direct mapped
 - L1D: 64KB, direct-mapped, write-through
 - L2: 256KB, direct-mapped, write-back
 - 4-word write buffer
 - ◆ Snooping implemented within a cluster (Illinois protocol, similar to MSI)
- General architecture of DASH: paper figure 1
- Sample 2x2 DASH system: paper figure 2

DASH Directories

- Directory controller (DC)
 - ◆ Directory memory corresponding to cluster's main memory portion
 - ◆ Initiates out-bound network requests and replies
- Pseudo-CPU (PCPU)
 - ◆ Buffers incoming requests and issues them on cluster bus
 - ◆ Mimics a CPU on behalf of remote processors (except for bus replies sent by DC)
- Reply Controller (RC)
 - ◆ Remote Access Cache (RAC) tracks outstanding requests by local processors
 - ◆ Receives and buffers corresponding replies from remote clusters
 - ◆ RAC snoops on bus
- Requests and replies sent on two different networks using wormhole routing (discussed later in course)
- Directory block diagram: paper figure 3

DASH Coherence Protocol

- Terminology
 - ◆ Local cluster: cluster containing the processor originating a request
 - ◆ Home cluster: cluster containing the main memory and directory for a given memory address
 - ◆ Remote cluster: Any cluster other than local and home clusters
 - ◆ Local memory: main memory associated with the local cluster
 - ◆ Remote memory: Any memory whose home is not the local cluster
- Invalidation-based protocol
 - ◆ Cache states: invalid, shared, and dirty
- Directory state (for all local memory blocks)
 - ◆ Uncached-remote: not cached by any remote cluster
 - ◆ Shared-remote: Cached, unmodified, by one or more remote clusters
 - ◆ Dirty-remote: Cached, modified, by one remote cluster
- Owning cluster for a block is the home cluster except if *dirty-remote*
- Owning cluster responds to requests and updates directory state

Read Requests

- Initiated by CPU load instruction
- If address in in L1 cache, L1 supplies data – otherwise, fill request sent to L2
- If address is in L2, L2 supplies data – otherwise, read request sent on bus
- If address is in the cache of another processor in the cluster or in the RAC, that cache responds
 - ◆ Shared: data transferred over the bus to requester
 - ◆ Dirty: data transferred over bus to requester, RAC takes ownership of cache line
- If address not in local cluster, processor retries bus operation, and request is sent to home cluster, RAC entry is allocated
- Requests to remote nodes explained in figure 4

Read-Exclusive Requests

- Initiated by CPU store instruction
- Data written through L1 and buffered in a write buffer
- If L2 has ownership permission, write is retired – otherwise, read-exclusive request sent on local bus
 - ◆ Write buffer is stalled
- If address is in “dirty” in one of the caches in the cluster or in the RAC
 - ◆ Owning cache sends data and ownership to requester
 - ◆ Owning cache invalidates its copy
- If address not in local cluster
 - ◆ Processor retries bus operation
 - ◆ Request is sent to home cluster
 - ◆ RAC entry is allocated
- Requests to remote nodes explained in figure 5

Other Implementation Details

- Writeback requests: When a dirty block is replaced
 - ◆ Home is local cluster: Write data to main memory
 - ◆ Home is a remote cluster: Send data to home which updates memory and state as “uncached-remote”
- Latency for memory operations: paper figure 6
- Exception conditions
 - ◆ Request to a dirty block of a remote cluster after it gave up ownership
 - ◆ Ownership bouncing back and forth between two remote clusters while a third cluster requests block
 - ◆ Multiple paths in the system lead to requests being received out of order
- Amount of information stored in directory affects scalability
 - ◆ For each memory block, DASH stores state and bit vector for other processors
 - ◆ For a more scalable system, overhead needs to be lower

The SGI Origin

- Cache coherent non-uniform memory access
- Up to 512 nodes
- Scalable Cray link network (hypercube)
- 1 or 2 R10000 MIPS processors per node
- Up to 4G bytes per node
- Node connects to a portion of the IO subsystem
- No snooping within node

Key Goals

- Scale to large number of processors
- Provide higher performance per processor
- Maintain cache-coherent globally addressable memory model
 - ◆ For ease of programming
- Entry level and incremental cost of the system lower than a high performance SMP

Origin Architecture

- Distributed shared memory (DSM)
- Directory based cache coherence
- Designed to minimize latency difference between local and remote memory
- Hardware and software provided to insure most memory references are local
- Origin block diagram: paper figure 1
- Cache coherence does not require in-order message delivery
- I/O subsystem is also distributed and globally addressable
- I/O can DMA to and from all memory in the system
- Cluster bus is multiplexed but is not a snoop bus
 - ◆ Reduce local and remote memory latency
 - Fewer processors on the bus
 - Remote request does not need to wait for snoop response

Origin Architecture (Cont.)

- Non-snoopy node bus tradeoff
 - ◆ Disadvantage: remote bandwidth needs to match local bandwidth, unlike in SMP node systems
 - ◆ Advantage: easier migration path for existing SMP software
- Page migration and replication insures most references are local
 - ◆ Memory reference hardware counters
 - ◆ Copy engine to copy at near peak memory bandwidth
- Rich synchronization primitives
- Fetch and op primitives are non cached and performed at memory
 - ◆ Useful in highly contended locks
 - ◆ HUB implements 4-way full cross bar between processors, memory and I/O-network
 - ◆ RAS features
 - ECC in external cache and memory
 - Faulty packets automatic retries
 - Modular design provides highly available hardware

Network

- Six ported router chip
- Fat-hypercube (paper figures 3 and 4)
- Low latency wormhole routing
- Four virtual channels per physical channel
- Congestion control to allow messages to adaptively switch between two virtual channels
- Support for 256 levels of message priority
- Increased priority via packet aging
- Automatic packet retries
- Software programmable routing tables

Cache Coherence Protocol

- Similar to DASH protocol but with significant improvements
 - ◆ MESI protocol is fully supported
 - Single fetch from memory for read-modify-writes
 - Permits processor to replace E block in cache without informing directory
 - Requests from processors that had replaced E blocks can be immediately satisfied from memory
 - ◆ Support of upgrade requests from S to E without data transfer

Configuration and Performance

- CPU Configuration
 - ◆ MIPS R10000
 - ◆ 195 MHz
 - ◆ 4-way out-of-order
 - ◆ 4 M byte L2 cache
 - ◆ Bus connected to the HUB chip
- Latency variation (paper table 4)
- High memory bandwidth (paper figures 11 & 12)
- Synchronization (paper figure 13)

Reading Assignment

- C. Seitz, “The Cosmic Cube”, Communications of the ACM, 1985 (Review)
- Homework 2 due Monday Oct 26 (deadline extended from this Wednesday)
 - ◆ Submission instructions posted on webpage