Cache Coherence in Manycores

Snooping Caches Definitions

Shared Memory

Bus or Ring

Ext. bus request

My local response

My bus response

Broadcast

Match

Dual ported

Update

write

snoop

Remember, “a” is memory word
In ownership protocol:
writer owns exclusive copy
What to Do When Cache Blocksize > 1 word

Suppose "a" is word address
State associated with entire cache block

Discuss: Effect of false sharing

Solving the Coherence Problem

- Small multicores
  > Software coherence
  > Snooping caches

- Manycores
  > Software coherence
  > Full map directories
  > Limited pointers
  > Chained pointers
  > Singly linked
  > Doubly linked
  > Limitless schemes
  > Hierarchical methods

We will study
  Coherence structures
  Coherence protocols
  Cache side state diagrams
  Directory side state diagrams
General directories:

- On read, track cached copy in directory
- Distribute directories with MEMs
  Directory bandwidth scales in proportion to memory bandwidth
- Most directory accesses during memory access -- so not too many extra network requests (except, write to read VAR)
- Manycores?
Directory controller state diagram for memory block

Cache state machine for directory scheme

Also, need to ack (implicitly or explicitly) every message from memory – we’ll see why later.
Next, let's look at several practical and scalable implementations of directory schemes.

Full-map directories do not scale. They can get large!
Limited directories: Exploit worker set behavior
- Rarely more than \(i\) processors share (\(i\) is small)
- Invalidate 1 if \((i+1)\)th processor comes along: \texttt{Dir\_NB}
- Insight: The set of 2 pointers can be managed like a fully-associative 2 entry cache on the virtual space of all pointers

But what do you do about widely shared data?

Set a broadcast invalidate bit and let more than \(i\) processors share: \texttt{Dir\_B}

What to do on a write?
- If \(B\) set, send broadcast invalidate

What is \texttt{Dir\_0B}?
Network

**Limitless directories:**

Limited directories Locally Extended through Software Support

- Trap processor when 3rd request comes
- Processor extends directory into memory (local cache)

Dir? → Dir\_nNB

Why might the performance be OK?

Any unique software opportunities here?

- 15 -

A crazy idea…. zero pointer limitless aka, all software directories:

- Trap processor, and let processor do usual directory operations in software
- Processor extends directory into memory (local cache)
- Zero hardware!

Software can even implement protocol best suited to each variable

Or it can change protocols on the fly (using, for example, competitive algorithms)

- 16 -
Performance

Weather modeling

Some Variants

Network

Directory caches:

Cache a few directory entries

No need to maintain directory entry for uncached blocks, or for private data (compiler/OS can tag private pages)
Chained directories: Simply different data structure for directory
- Link all cache entries
- On read, add new entry to end of list

What to do on write?

Chained directories: What to do on a write?
- Invalidate all entries on write
Chained directories: What to do on a write?
- Invalidate all entries on write
- Only written block added to new list

Problems?
- But longer latencies
- Also more complex... why?

Chained directories:
- Must handle replacements of elements in chain due to misses!

Invalidate rest
Another idea...

Doubly linked chains

Even more complex

Hierarchical Protocols - E.g. KSR, Kendall Square Research Inc. (actually had rings...)

Multicores will probably initially scale this way as we look to do coherence across two chips
Hierarchical cache

First read

Hierarchical cache

Subsequent read
Hierarchical cache disks

MEMORY

Hierarchical Write

MARKED

disks

MEMORY

Marked as written

Hierarchical Subsequent writes