

18-742

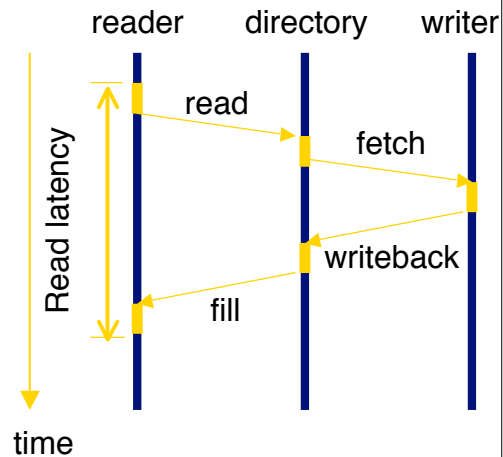
Lecture 17

Coherence Optimization

Spring 2005

Prof. Babak Falsafi

<http://www.ece.cmu.edu/~ece742>



Slides developed in part by Profs. Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith, and Singh of University of Illinois, Carnegie Mellon University, University of Wisconsin, Duke University, University of Michigan, and Princeton University.

Readings

Chapter 9 of Culler & Singh

Reader 5:

- A.-C. Lai and B. Falsafi, *Selective, accurate, and timely self-invalidation using last-touch prediction*, ISCA 2000.
- M. M. K. Martin, M. D. Hill, and D. A. Wood, *Token Coherence: Decoupling Performance and Correctness*, ISCA 2003.
- C. Amza, et al., *TreadMarks: Shared Memory Computing on Networks of Workstations*, IEEE Computer 29(2): 18-28, 1996.

Announcements

Status report:

- Actual preliminary results
- Not, infrastructure bugs

Course evaluations:

- Feedback given on graded homework

Types of Coherence Optimization

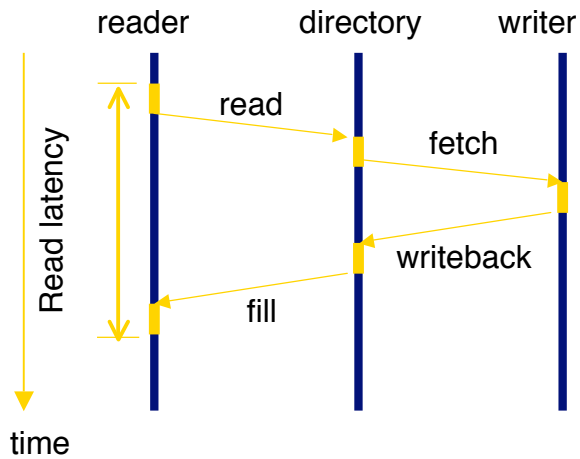
Memory consistency models:

- Hide store latency (via annotation, speculation, or both)

Other optimization types:

- Hide latency
 - Prefetching
 - Forwarding
- Coherence optimization
 - Reduce traffic
 - Overlap transitions
 - » Request forwarding
 - » Collecting acknowledgements at writer
 - Reduce transitions
 - » Sharing prediction
 - Hide transitions
 - » Dynamic self-invalidation

Example Shared Read in DSM

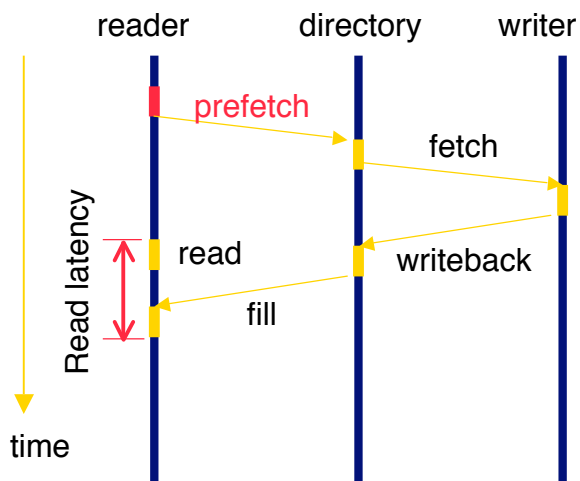


(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

5

Example Prefetched Shared Read in DSM



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

6

Prefetching Revisited

Same issues as uniprocessor:

- “what” to prefetch is difficult
- “when” to prefetch is even more difficult
 - Distances are larger
 - May pollute caches or may prefetch too late (as before)
 - But, may also take data away from current user (multiprocessor)

Can Forward too:

- same problems as prefetching
- “who” and “When”

Can be done in SW, HW or both

Reduce Traffic

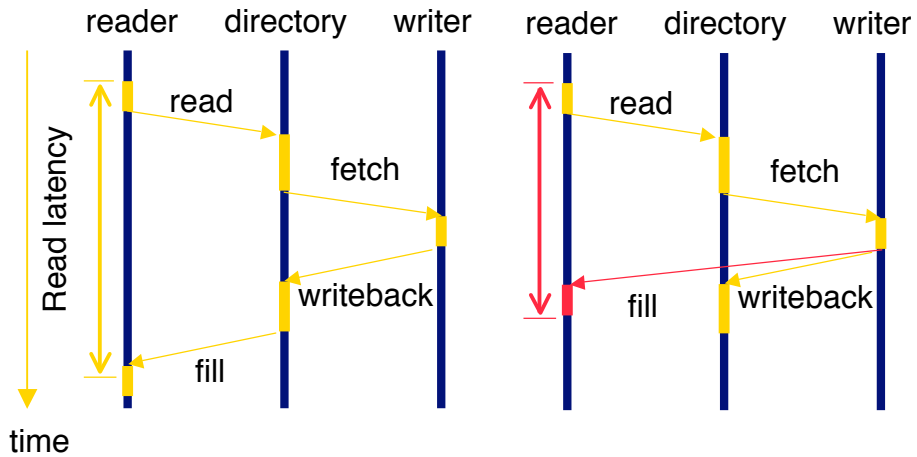
Enhanced caching:

- R-NUMA

Reduce false sharing:

- Pad data structures (what is wrong with this?)
- Spatial pattern prediction [Chen et al.]
- Coherence decoupling
 - Value prediction assuming old (stale) values are still in the cache

Overlapping Transitions: Request Forwarding



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

9

Request Forwarding

Also called a “3-hop” rather than a “4-hop” protocol

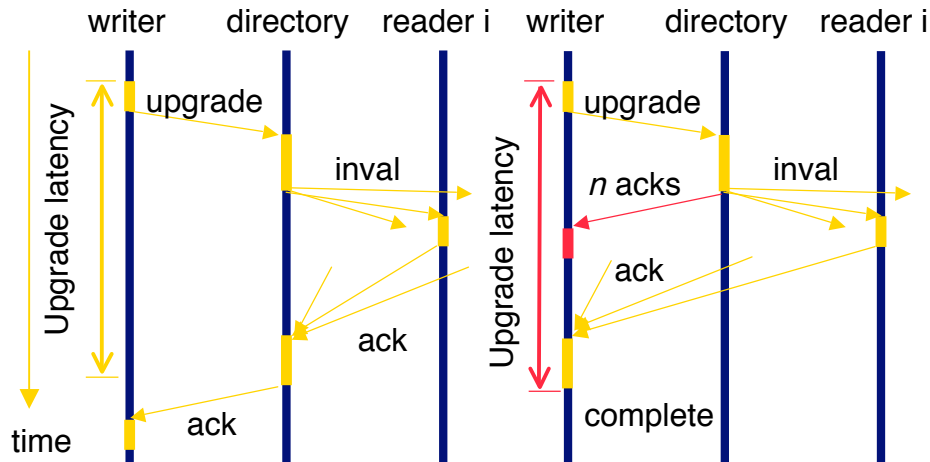
- Implemented in Stanford DASH
 - Got it right in SGI Origin 2000
- Causes deadlock if not implemented correctly
- Why?

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

10

Overlapping Transitions: Optimizing Acknowledgements



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

11

Optimizing Acks and Invals

Collecting Acks at Writer

- Distributes the load
- Obviates the need for bookkeeping at directory
- E.g., Piranha CMP

Eager response to upgrade:

- Directory can respond and inval in background
- Has memory order implications
 - Stores commit but are not completed!
- Implemented in Compaq/HP AlphaServer GS320

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

12

Coherence Optimization: Example Migratory Sharing

Proc 1	Proc2	Proc 3
Read X Write X		
		Read X Write X
	Read X Write X	

(C) 2005 Babak Falsafi from Adve, Falsafi,
Hill, Lebeck, Reinhardt, Smith & Singh

18-742

13

Migratory Sharing Transitions

Every Read/Write pair results in:

- **A remote read miss**
- **Followed by an upgrade request**

Coherence FSM can keep track of this

At the directory:

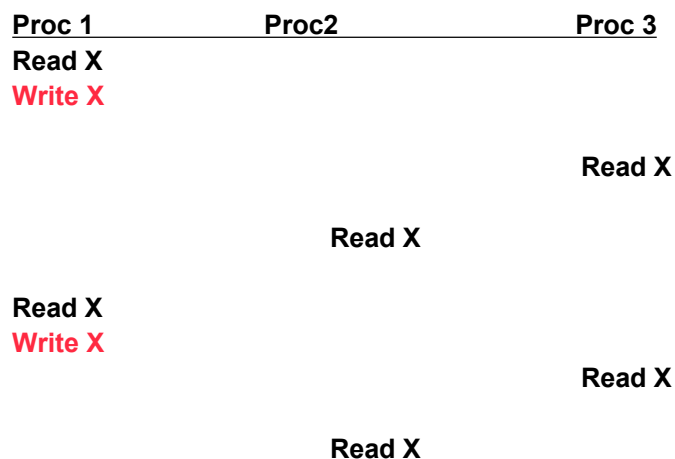
- **Two back-to-back requests for Read/upgrade by same proc**
- **Make a transition to “migratory Modified” state**
- **Upon a read, invalidate current copy**
- **Return a writable copy (i.e., M state)**

(C) 2005 Babak Falsafi from Adve, Falsafi,
Hill, Lebeck, Reinhardt, Smith & Singh

18-742

14

Coherence Optimization: Example Producer/Consumer Sharing



(C) 2005 Babak Falsafi from Adve, Falsafi,
Hill, Lebeck, Reinhardt, Smith & Singh

18-742

15

Producer/Consumer Transitions

Simple optimization:

- Upon read miss, “downgrade” rather than invalidate “writer”
 - Distinguish this transition from the migratory case
- Results in one upgrade from writer, followed by reads by others

More sophisticated optimizations:

- Keep track of prior readers
- Forward to all readers upon the first read

(C) 2005 Babak Falsafi from Adve, Falsafi,
Hill, Lebeck, Reinhardt, Smith & Singh

18-742

16

Shortcomings of Coherence Optimizations

Optimizations built directly into coherence transitions

- e.g., migratory sharing in SGI Origin
- Not a great idea!
- Coherence protocols are extremely complex machines
- Hard to verify even basic protocols
- Each optimization -> extra complexity -> state explosion!

Plus

- Must only target simple sharing patterns
- Can only learn/optimize one sharing pattern at a time
 - Data structures may exhibit multiple patterns throughout the program

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

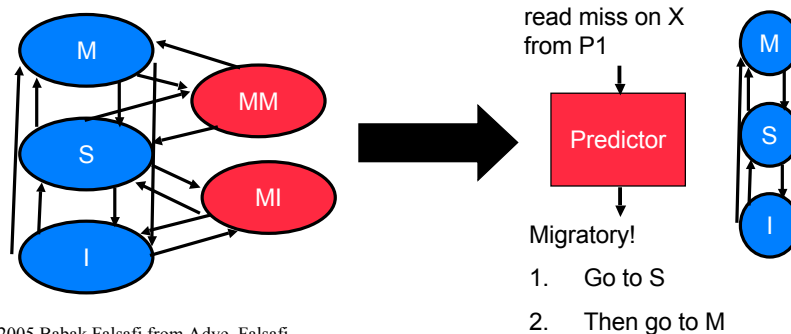
18-742

17

Recent Proposals: Table-Based Predictors

Decouple predictor from protocol

- + learn multiple sharing patterns simultaneously
- + protocol hints → no impact on state machine
- may require large storage overhead

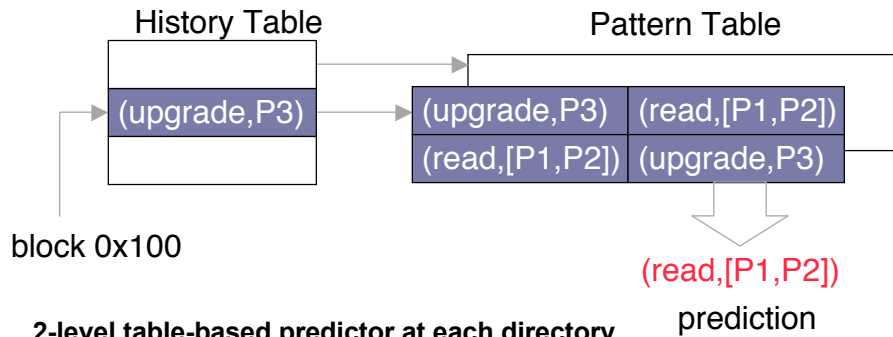


(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

18

Memory Sharing Predictor (MSP) [Lai et al.]



2-level table-based predictor at each directory

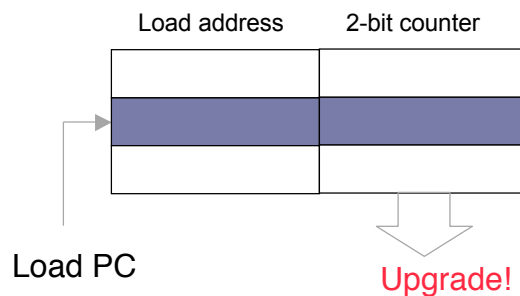
- Keeps a history of prior messages
- For each history, keeps a sharing outcome
- E.g., an upgrade by P3 led to reads by P1 and P2

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

19

Upgrade Predictor [Kaxiras et al.]



PC-indexed table-based predictor

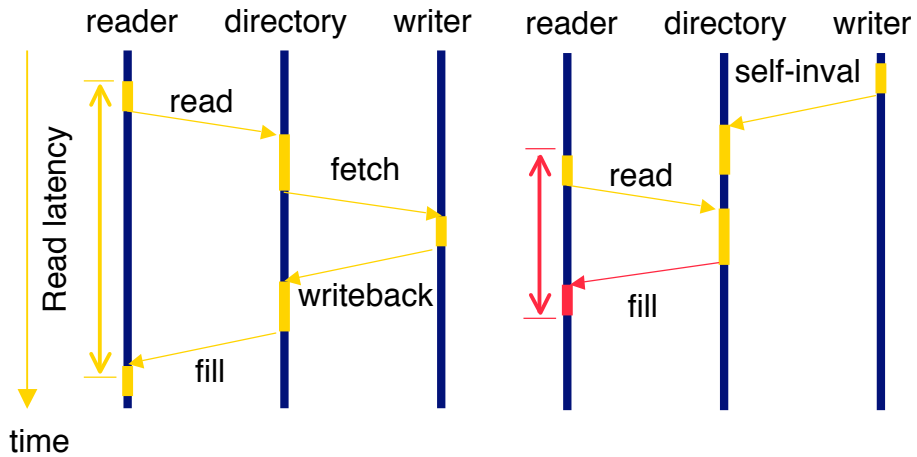
- On a load, store address
- If count saturated, upgrade
- On a subsequent store, increment count
- On invalidations, decrement count

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

20

Hiding Transitions: Dynamic Self-Invalidation



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

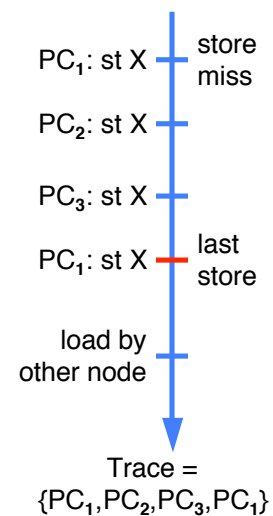
18-742

21

Trace-based Downgrade Prediction

Predicts last store to modified cache block

- **Records a trace of stores to a block**
 - From store miss to last store
- **When trace recurs, predict last store**
- **Timely downgrades**
 - Triggered immediately on last store
- **Good prediction: 3-hop → 2-hop**
- **Misprediction: store miss**
 - Relaxed memory model hides penalty



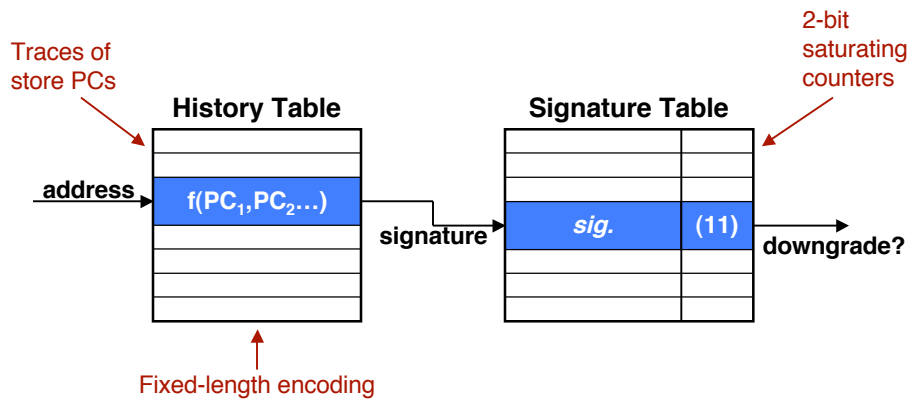
(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

22

Predictor Structure (Recall Dead-Block Predictor)

- 2-level predictor, derived from Last Touch Predictor
 - [Lai & Falsafi]



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

23

Methodology

- Trace-driven simulator using SimFlex
- Memory traces from full-system simulation
- Solaris: 16 CPUs Linux: 8 CPUs

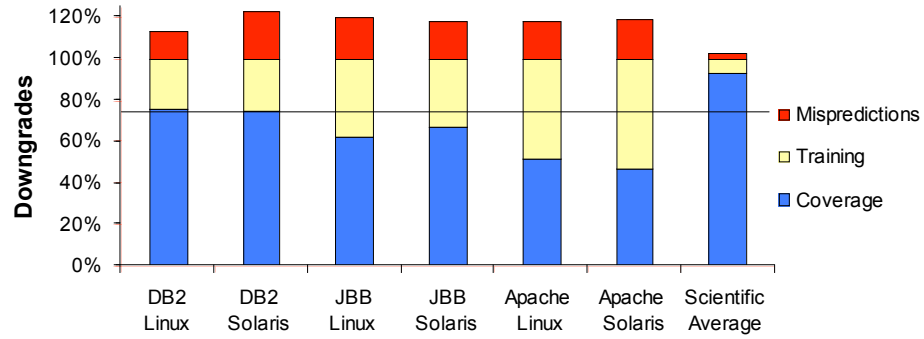
Application	Configuration
DB2 Linux	TPC-C, 100 warehouses
DB2 Solaris	TPC-C, 100 warehouses
JBB Linux	SPECjbb2000, 8 warehouses
JBB Solaris	SPECjbb2000, 16 warehouses
Apache2 Linux	SPECweb99, Apache 2.0.48
Apache1 Solaris	SPECweb99, Apache 1.3.27
Scientific Apps	barnes, em3d, moldyn, ocean, water

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

24

Predictor Results



- 47% - 76% coverage on commercial workloads
- Mispredictions ($\leq 22\%$) can be overlapped
- Scientific results confirm previous work

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

25