Using Prediction to Accelerate Coherence Protocols

Shubhendu S. Mukherjee *(Shubu)*[*] & Mark D. Hill

Computer Sciences Department
University of Wisconsin-Madison
{shubu,markhill}@cs.wisc.edu
http://www.cs.wisc.edu/~{shubu,markhill}

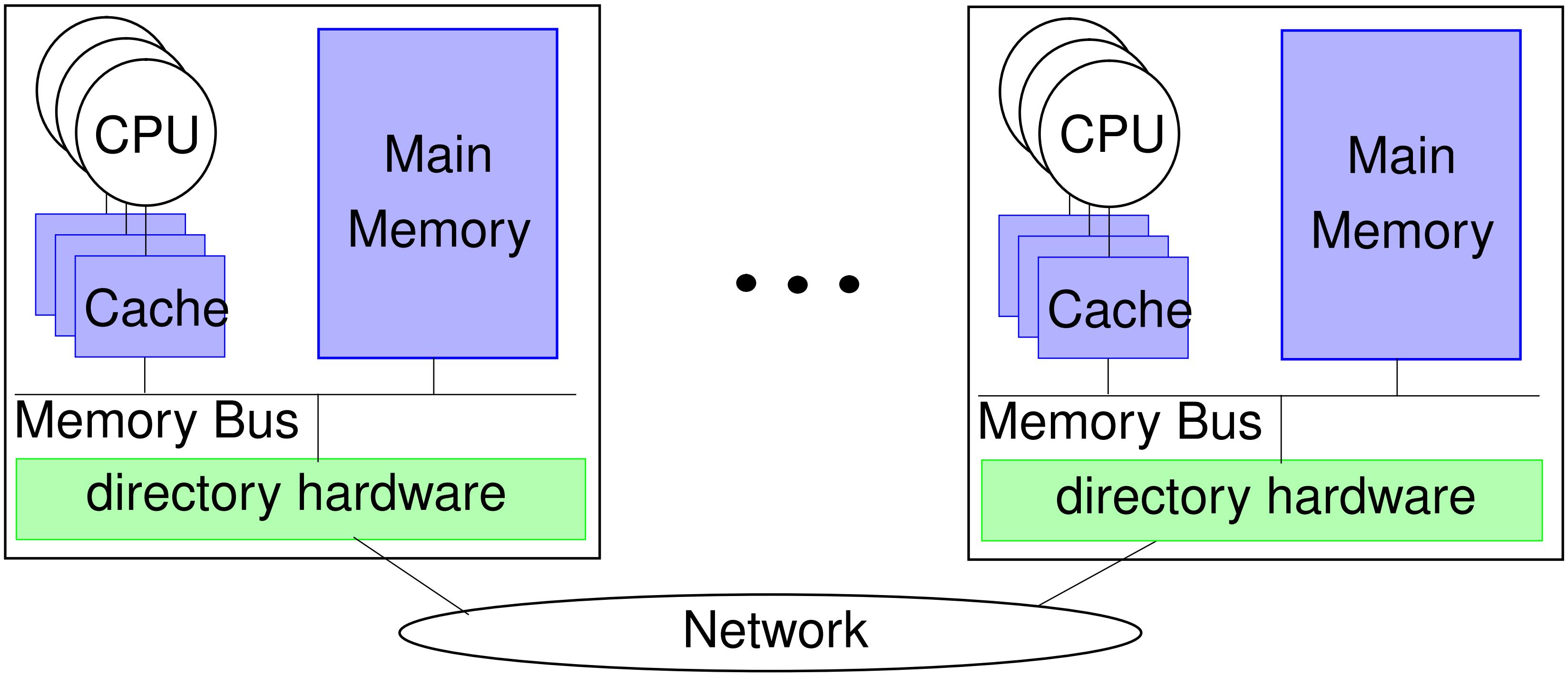
Motivation & Overview

Cosmos' Structure

Cosmos Results

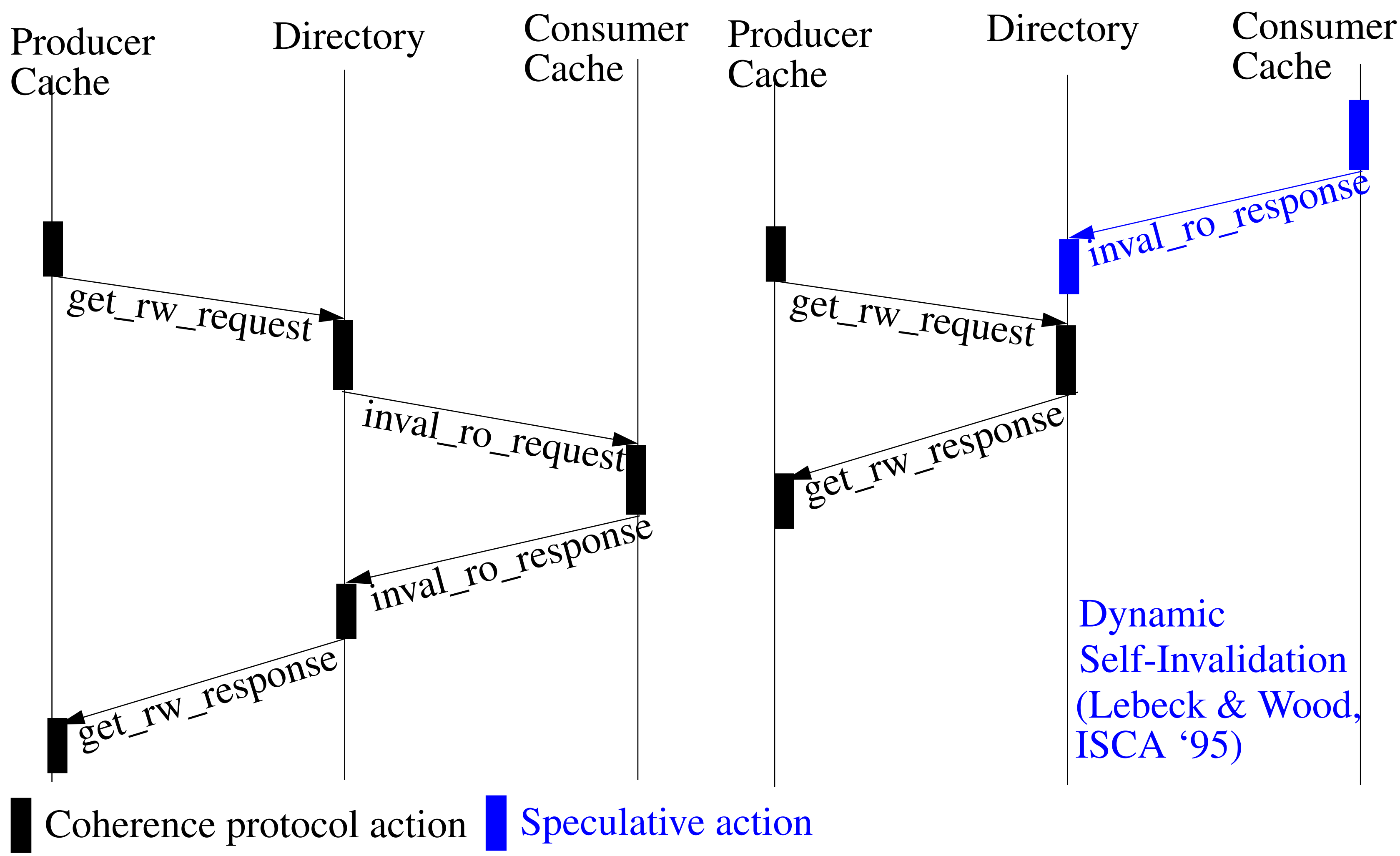
*. Current Address: Digital Equipment Corporation, Shrewsbury

Distributed Shared-Memory Machine



- Memory is physically distributed for scalability
- Per-CPU caches cache remote memory
- Cache coherence via directory protocols

Reduce Directory Protocol Latency using Prediction



Directed Predictors

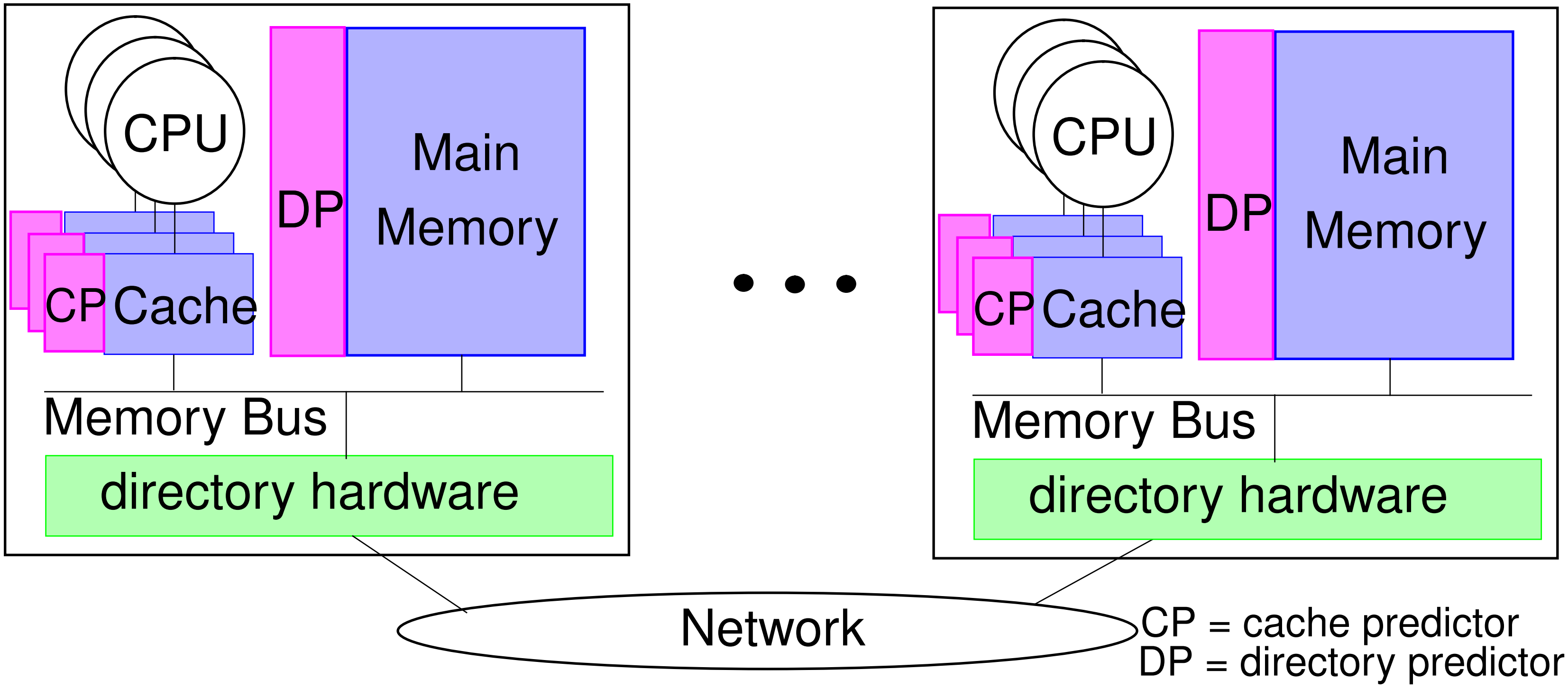
Many Examples:

- Read-modify-write in SGI Origin (Laudon & Lenoski, ISCA ‘97)
- Scalable Coherent Interface (SCI)’s pairwise sharing
- Protocols optimized for migratory sharing (Cox & Fowler, Stenstrom, et al. ISCA ‘93)
- Dynamic Self-Invalidation (Lebeck & Wood, ISCA ‘95)
- Competitive Update (Karlin, et al. Algorithmica, ‘88)
- Half-migratory optimization
- Other examples include compiler-directed prediction

Can we have a general predictor? => Cosmos

+ easier to compose multiple predictors
+ discover & adapt to application-specific patterns
- more hardware

Cosmos: A General Predictor



Cosmos predictors for both cache & directory

Predictor issues

- what message to predict? .... this talk
- how to integrate with real system? .... NOT in this talk

Cosmos Overview

Given

- cache block address
- history of incoming coherence messages for the cache block (i.e. source processor & message type tuples)

Cosmos predicts

- the next incoming coherence message for the cache block

Cosmos' Structure

- two-level adaptive predictor
- resembles Yeh & Patt's *PAp* branch predictor (ISCA '92)

Cosmos' Prediction Accuracy

- 62 - 93% for five parallel scientific applications

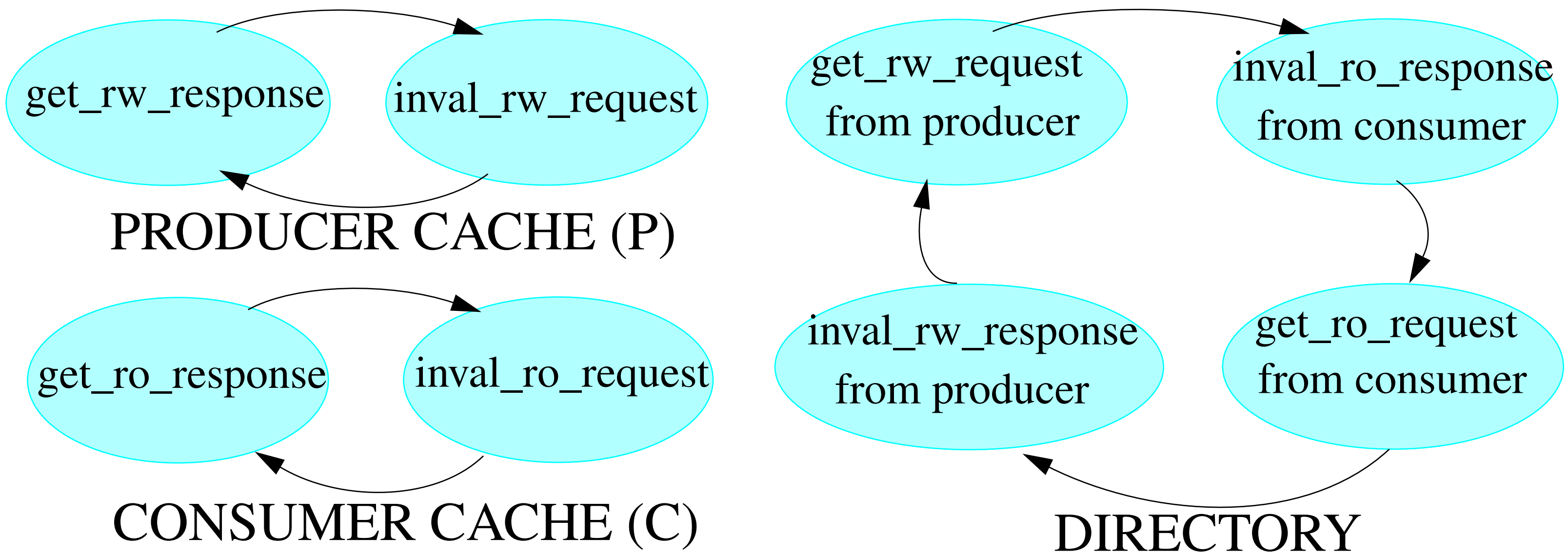
Outline

Motivation & Overview

Cosmos' Structure

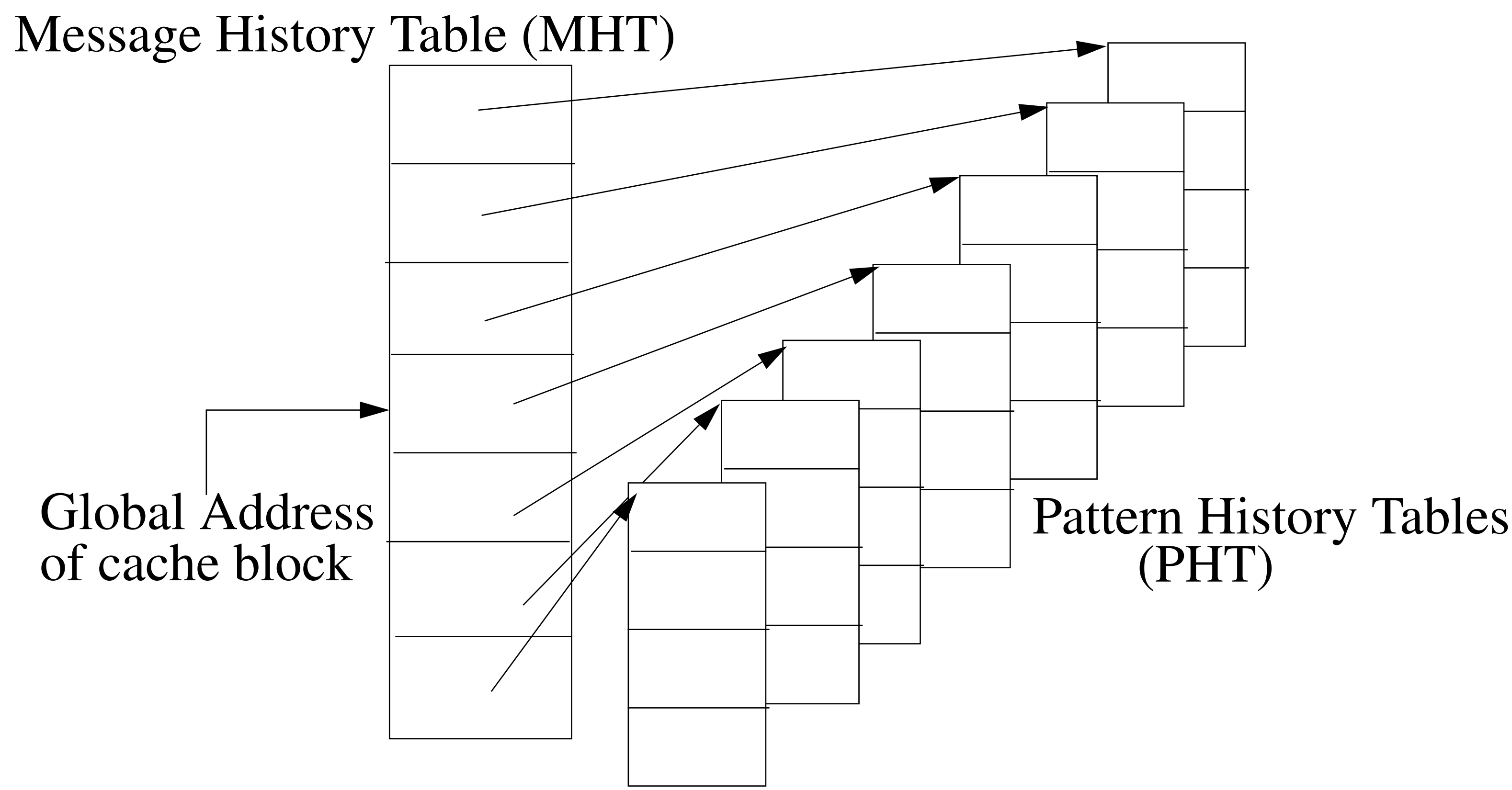
Cosmos Results

Producer-Consumer Sharing Pattern



Cache blocks have predictable message signatures

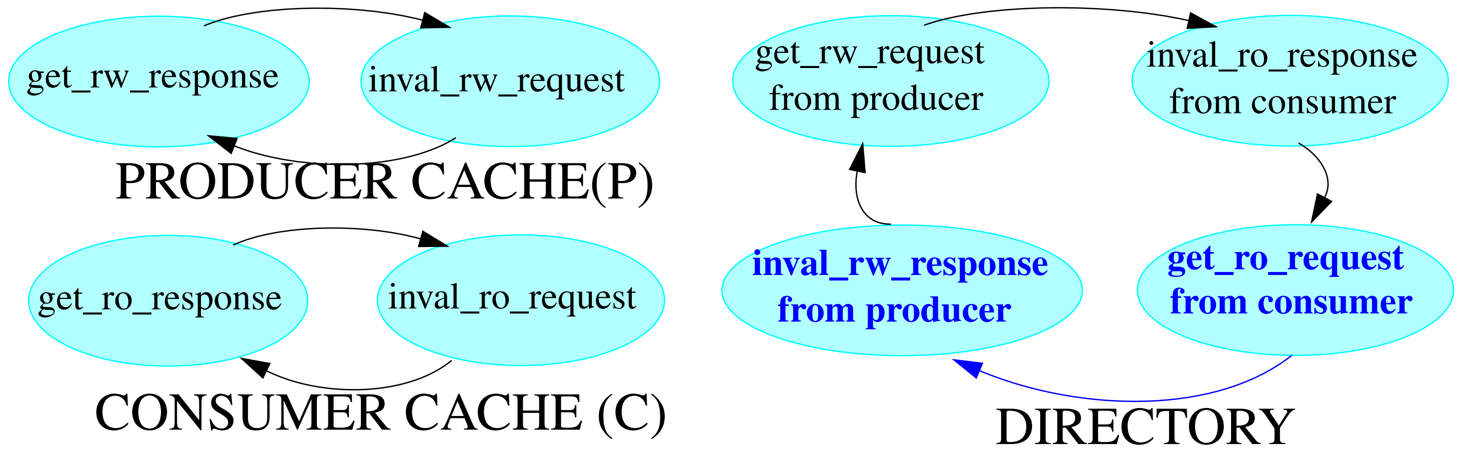
Cosmos' Basic Structure

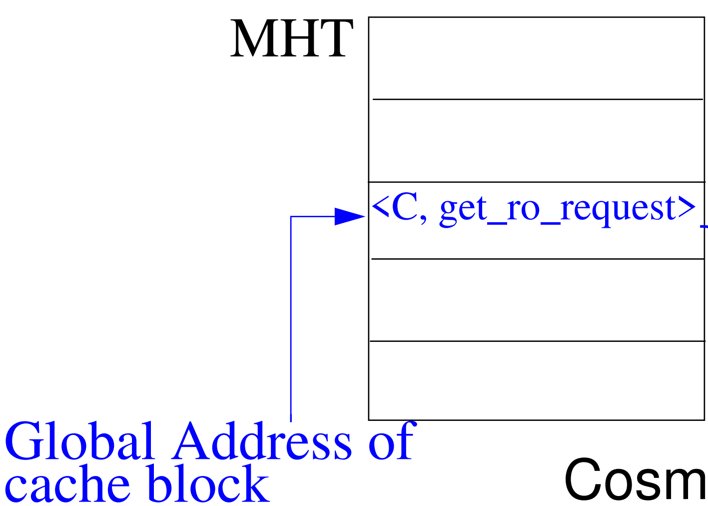


Parameterized by "depth" of MHT and "filters" for PHT

(Reminiscent of Yeh and Patt's PAp branch predictor)

Cosmos’ Entries for Producer-Consumer Signature





PHT

Index	Prediction
<P, get_rw_request>	<C, inval_ro_response>
<C, inval_ro_response>	<C, get_ro_request>
<C, get_ro_request>	<P, inval_rw_response>
<P, inval_rw_response>	<P, get_rw_request>

Cosmos at the directory

Motivation & Overview

Cosmos' Structure

Cosmos Results

Evaluation Methodology

Traces of coherence messages

Simulator

- Wisconsin Wind Tunnel II (Mukherjee, et al. PAID '97)

Simulated coherence protocol = Wisconsin Stache

- Full-map
- Simple COMA (main memory used as software cache)
- Reinhardt et al. ISCA '94

Simulated benchmarks

- appbt .... NAS
- barnes .... SPLASH II
- dsmc..................................... Univs. of Maryland & Wisconsin
- moldyn.................................... Univs. of Maryland & Wisconsin
- unstructured .... Univs. of Maryland & Wisconsin

Cosmos' Base Prediction Rate

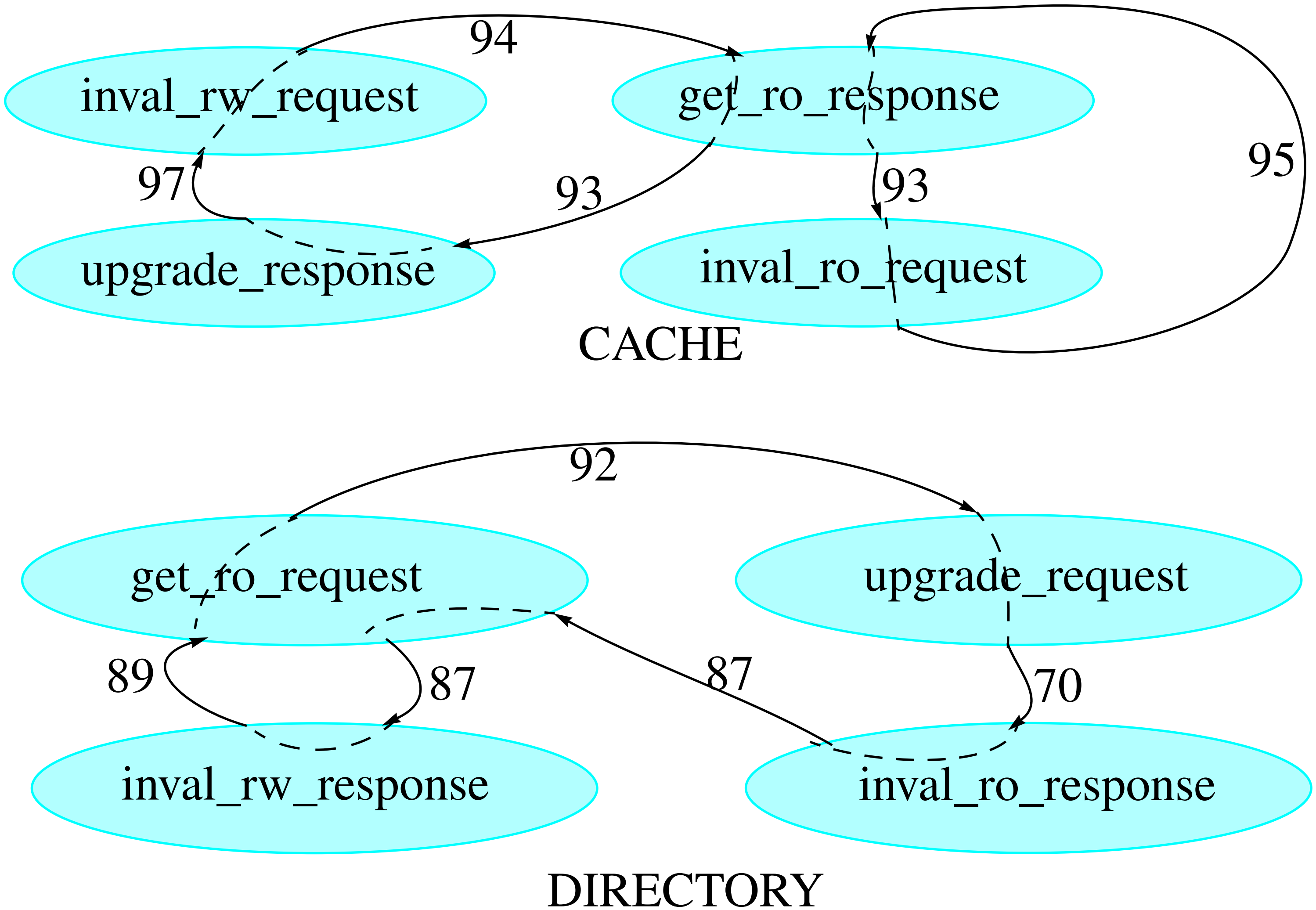
	appbt	barnes	dsmc	moldyn	unstructured
Cache	91%	80%	94%	92%	85%
Directory	77%	42%	73%	79%	65%
Overall	84%	62%	84%	84%	74%

Overall Accuracy = 62-84% (Base)

Low accuracy for barnes

- reassignment of logical data structures to different memory addresses

Example Signatures: Appbt



Increasing Cosmos' Accuracy

Depth of MHR	appbt	barnes	dsmc	moldyn	unstructured
1	84	62	84	86	74
2	85	69	86	86	88
3	85	69	93	85	89
4	85	69	93	84	92

Overall prediction accuracy = 62 - 93%

Other techniques

- filters (e.g., J. Smith's saturating counter)
- subdividing coherence message stream (suggested by Sohi)
  - available in Mukherjee, Ph.D. Thesis, May 1998
  - ftp://ftp.cs.wisc.edu/wwt/Theses/mukherjee-1side.ps

Cosmos' Memory Overhead

Depth of MHR	*appbt*		*barnes*		*dsmc*		*moldyn*		*unstructured*	
	Ratio	**Ovhd**	**Ratio**	**Ovhd**	**Ratio**	**Ovhd**	**Ratio**	**Ovhd**	**Ratio**	**Ovhd**
1	1.2	5.4%	3.8	13.5%	0.8	3.9%	0.8	4.0%	1.7	6.8%
2	***1.4***	***9.6%***	***6.9***	***35.4%***	***0.4***	***5.1%***	***1.1***	***8.3%***	***2.1***	***12.8%***
3	1.9	16.4%	9.3	63.0%	0.3	6.7%	1.6	14.9%	2.8	21.9%
4	2.6	26.5%	10.9	91.8%	0.3	8.9%	2.0	21.6%	3.4	33.0%

Ratio = total number of PHT entries / total number of MHT entries

Ovhd = average memory overhead per 128-byte block

For MHR depth = 2

overhead < 13% for all, except barnes

overhead = 35% for barnes

Summary & Future Work

Cosmos Predictor

- predicts next coherence message for a cache block
- uses history information

+ simpler than composition of multiple directed predictors

+ adapts dynamically to application-specific coherence streams

- requires more hardware than directed predictors

Cosmos' Prediction Accuracy

- 74 - 93% for four applications
- 62 - 69% for barnes (reassignment of logical data structures)

Future Work

- integrate Cosmos with a coherence protocol
- evaluate Cosmos with proper timing simulation

Comparison with Yeh & Patt's PAp Branch Predictor

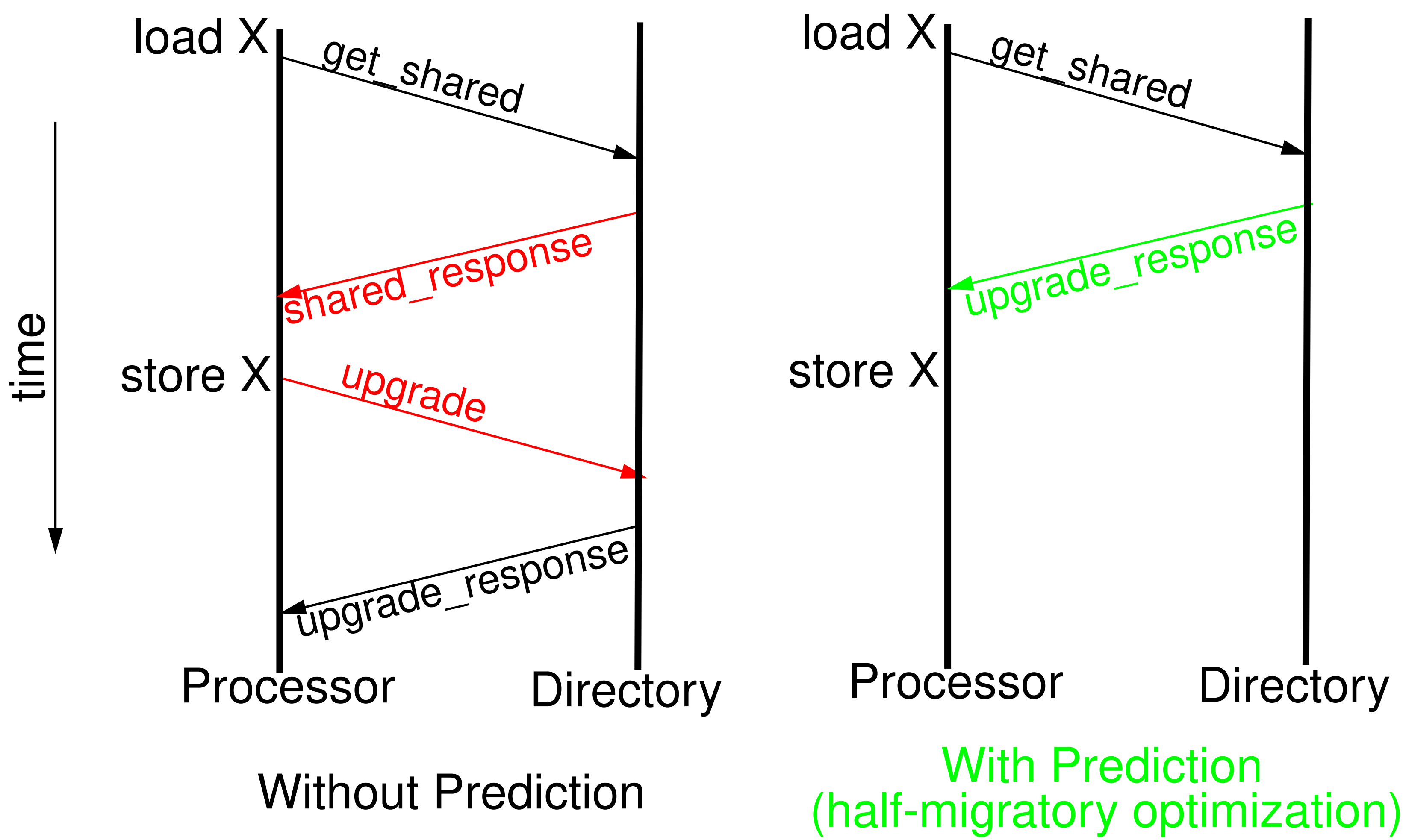
Similarities

- Cosmos has a two-level structure like PAp

Differences

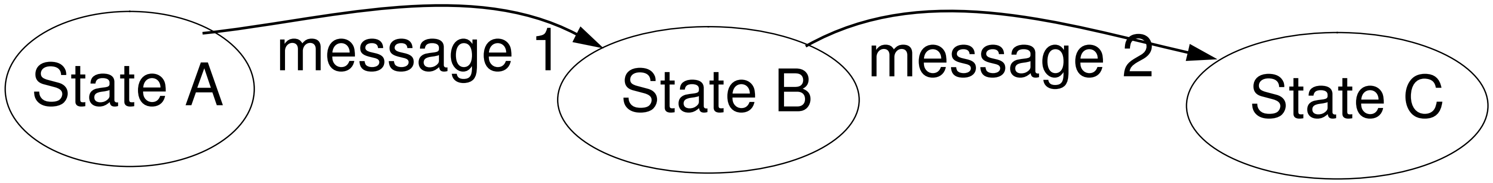
- Cosmos' first-level table (MHT) indexed by cache block address
- Cosmos' prediction alternatives > 2
- Cosmos is a multi-bit predictor

Half-Migratory Optimization



State- vs. Message-Based Prediction

Equivalent in functionality



message 1 -> message 2 $\equiv$ state B -> state C

Message-based prediction

+ fewer bits to store in prediction tables

    message requires processor id and few types

    state can require sharer ids, counter, state (incl. transition states)

+ naturally fits in if prediction is triggered by messages

- counter-intuitive?

Integrating Cosmos with Protocols

Mapping predictions to actions

- example: return “exclusive” block on read-modify-write prediction

Performing predictions at the “right” time

- example: on receiving request for shared block for read-modify-write prediction

Handling mis-predictions

detection

- example: did correct message arrive?

recovery

- no recovery if protocol moves between “legal” states
- recovery needed if

  coherence protocol moves to future (not exposed to processor)

  both coherence state and processor state move to future

Comparison with Directed Optimizations

Directed optimizations

- optimize for specific sharing pattern
- usually have a predictor built into protocol states

Examples

- Lebeck & Wood's dynamic self-invalidation
- Stenstrom, et al. and Cox and Fowler's migratory protocols

Cosmos

\+ single predictor

composing/extending multiple optimizations easier

\+ discovers application-specific patterns not know *a priori*

composition of migratory and producer-consumer in *unstructured*

\- more hardware?

\- can it be successfully incorporated into a working protocol?