LogTM-SE: Decoupling Hardware Transactional Memory from Caches

Luke Yen, Jayaram Bobba,
Michael R. Marty, Kevin E. Moore, Haris Volos,
Mark D. Hill, Michael M. Swift, and David A. Wood

Multifacet Project (www.cs.wisc.edu/multifacet)
Computer Sciences Dept., Univ. of Wisconsin-Madison

Executive Summary

- Hardware Transactional Memory (HTM) Fast
  - HW handles old/new versions (e.g., write buffer)
  - HW handles conflict detection (R/W bits & coherence)
- But Closely Coupled to L1 cache
  - On critical paths & hard for SW to save/restore

- Our Approach: Decoupled, Simple HW, SW control
- LogTM Signature Edition (LogTM-SE)
  - HW: LogTM's Log + Signatures (from Illinois Bulk)
  - SW: Unbounded nesting, thread switching, & paging

Outline

- Motivation
  - How HTMs accelerate TM
  - HTM Issues
  - Our Approach
  - Logs for version management
  - Signatures for conflict detection
- LogTM Signature Edition (LogTM-SE) Hardware
- LogTM-SE in Operation
- Conclusions and future work

How HTMs Accelerate TM

- Version Management
  - Save old values (for abort) & new values (for commit)
  - Write buffer, cache incoherence, overflow structures
- Conflict Detection
  - Record read/write sets & detect overlaps
  - Read/Write (R/W) bits on cache blocks & coherence

Some HTM Issues

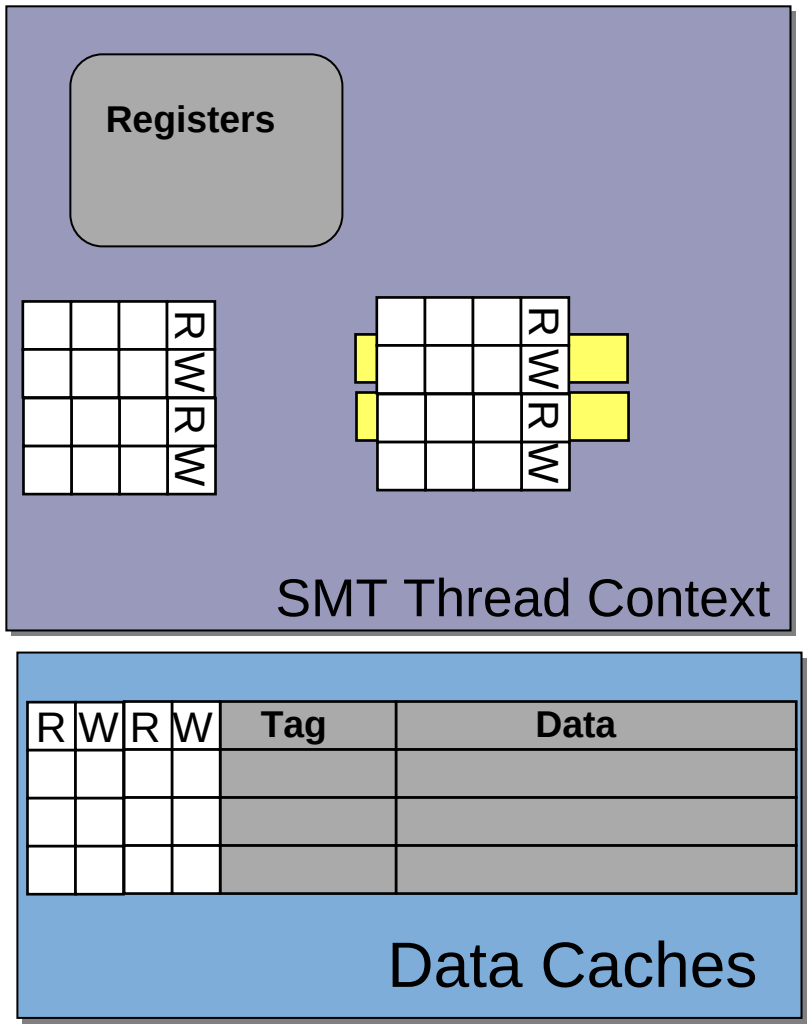
- R/W bits in precious L1 cache design
- Replicate R/W bits for SMT?
- Replicate again for (bounded) nesting?
- Save/restore R/W for thread switch?
- Modify for paging (virtual page moves)?

Our Approach

- Decoupled: Decouple HTM state from L1 caches
- Simple HW: Keep HW state simple & SW accessible
- SW Control: Have SW manage rare, complex events
- (Apply classic “systems” principles to HTMs)

Decoupling R / W bits from Cache

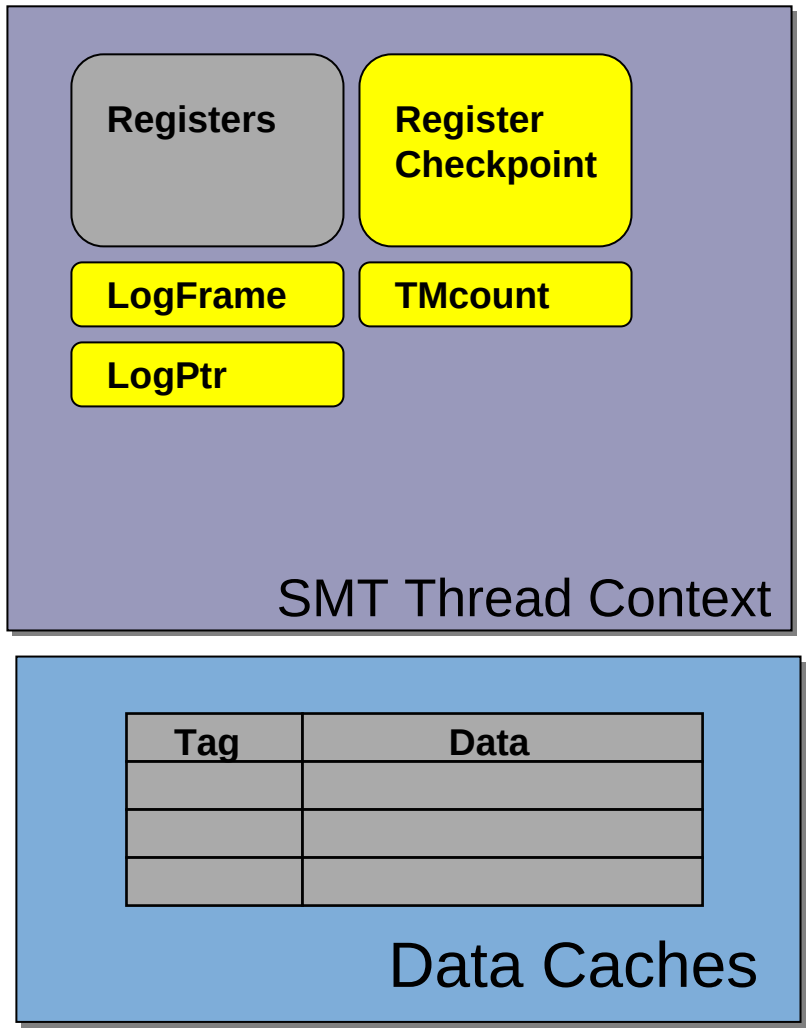
BEFORE:



Version Management

- Save old values (for abort) & new values (for commit)
- Use LogTM's Log
  - Before writing new values into memory, HW writes old values (and their virtual addresses) into log
  - Allocated per-thread in virtual memory (like pthread's stacks)
  - Log exposed to SW & not tied to a processor
- Why?
  - Decoupled, Simple HW, SW Control

Version Management Processor Hardware



Conflict Detection

- Record read/write sets & detect overlaps
- Adapt Signatures from Bulk [Ceze et al., ISCA 2006]
  - Over-approximate read/write sets in per-proc. Bloom Filters
  - Check signatures on coherence events (unlike Bulk)
  - Replicate for SMT
  - Exposed to SW for nesting, thread switching, etc.
- Why?
  - Decoupled, Simple HW, SW Control

Signature Operation

Program:

xbegin

LD A

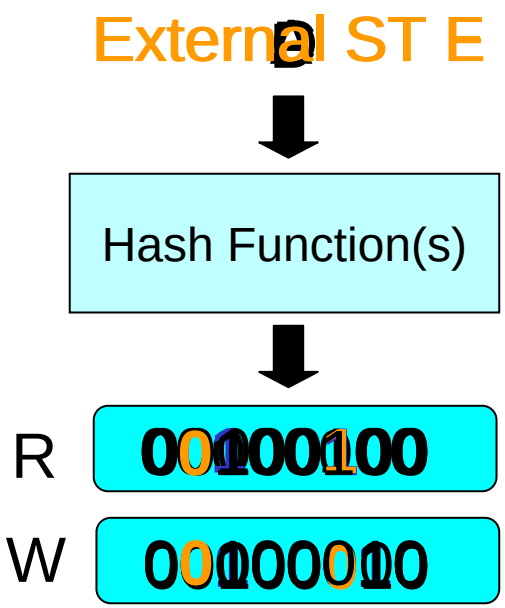
ST B

LD C

LD D

ST C

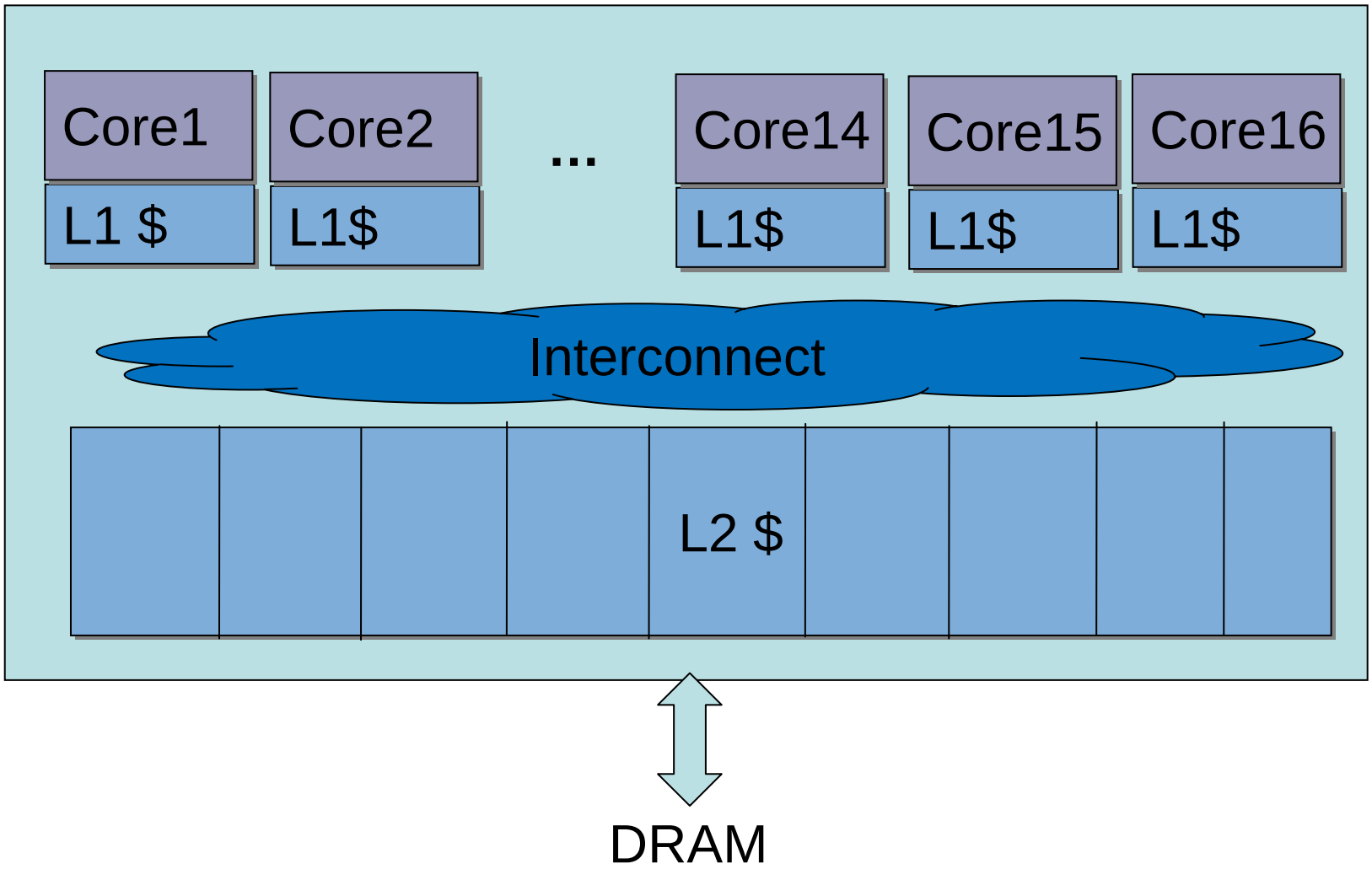
…



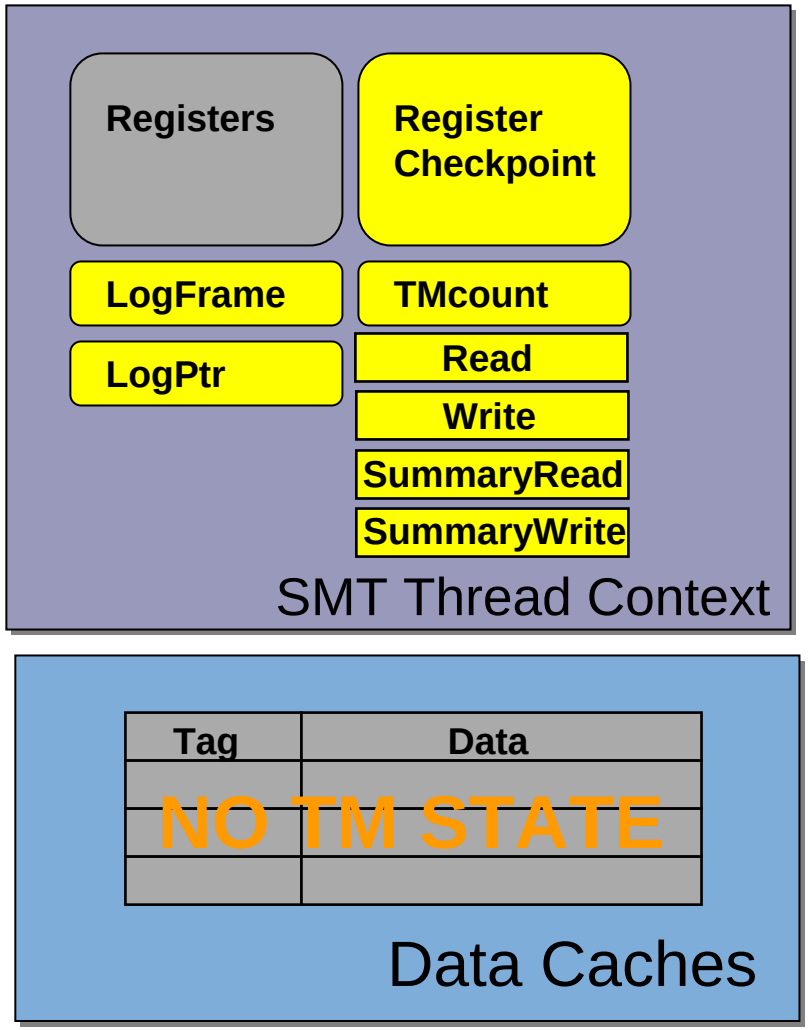
Outline

- Motivation
- LogTM Signature Edition (LogTM-SE) Hardware
  - Single-CMP system
  - Processor hardware
  - Experimental Results
- LogTM-SE in Operation
- Conclusions and future work

Single-CMP System



LogTM-SE Processor Hardware

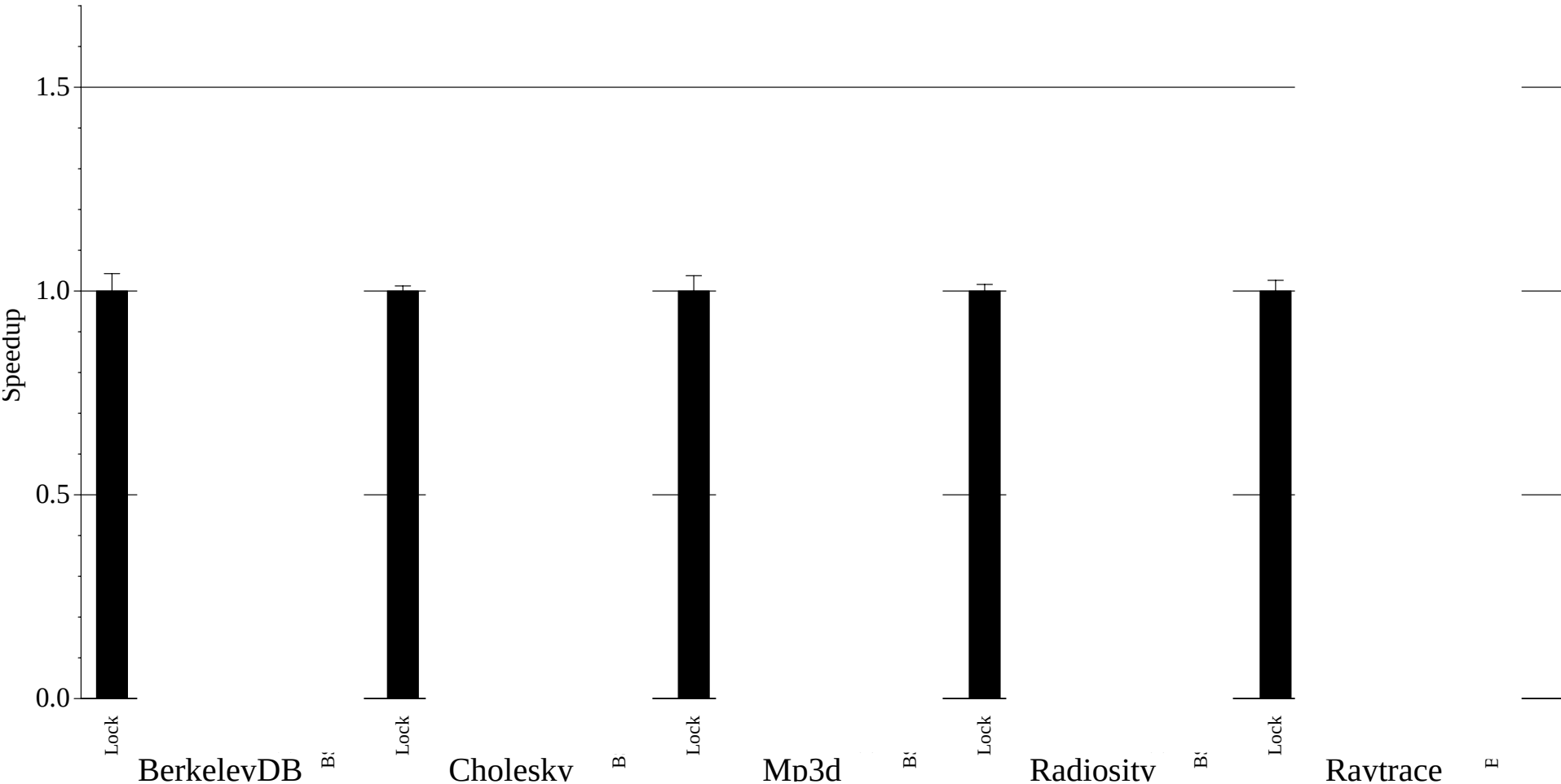


- Segmented log, like LogTM
- Track R / W sets with R / W signatures
  - Over-approximate R / W sets
  - Tracks physical addresses
  - Summary signature used for virtualization
- Conflict detection by coherence protocol
  - Check signatures on every memory access for SMT

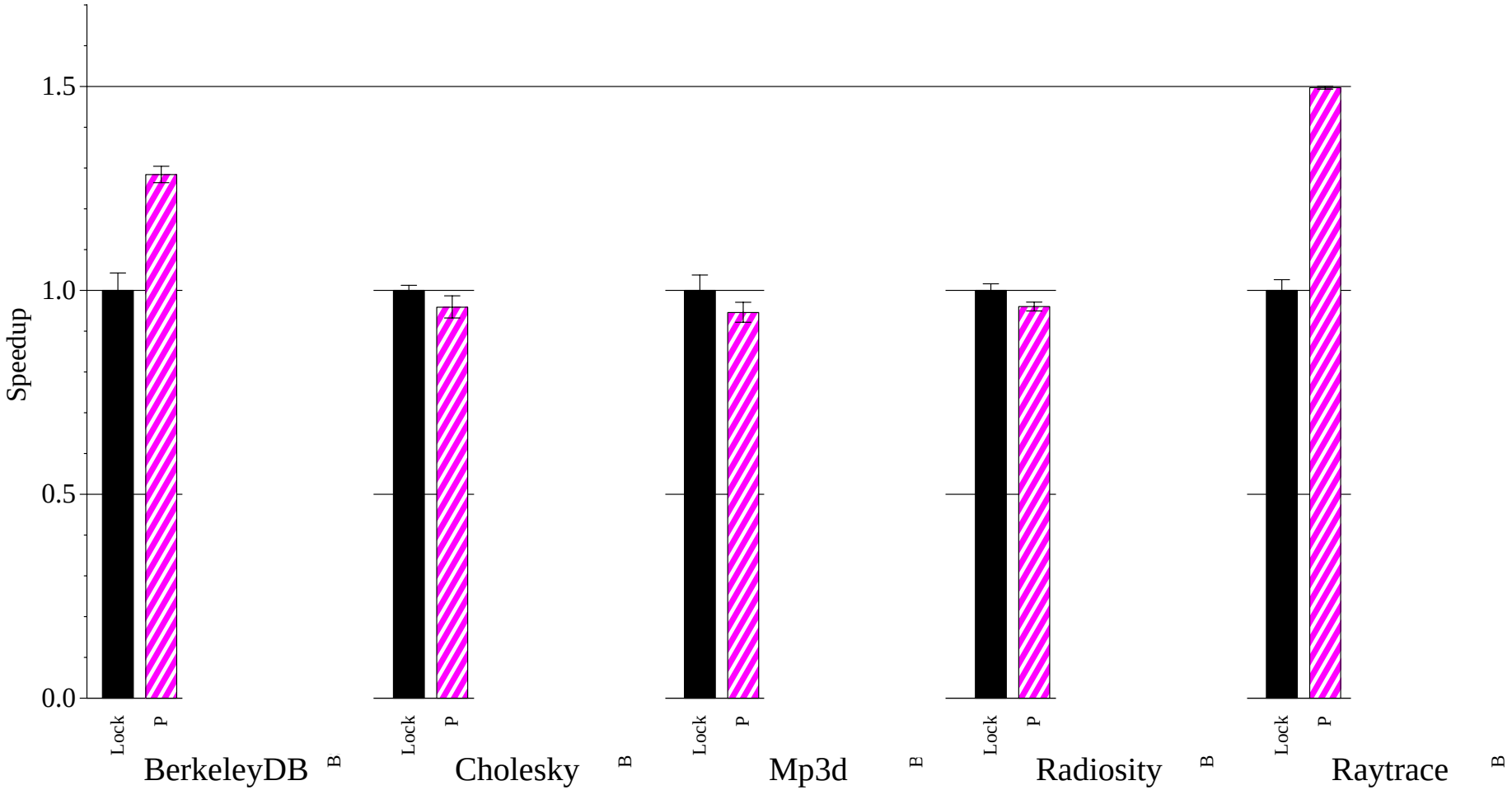
Experimental Methodology

- Infrastructure
  - Virtutech Simics full-system simulation
  - Wisconsin GEMS timing modules
- System
  - 32 transactional threads (16 cores x 2 SMT threads/core)
  - 32kB 4-way L1 I and D, 64-byte blocks, 1cycle latency
  - 8MB 8-way unified L2, 34 cycle latency
  - L2 directory for coherence, maintains full sharer bit vector
- Workloads
  - Radiosity, Raytrace, Mp3d, Cholesky **Berkeley DB**

Lock Results

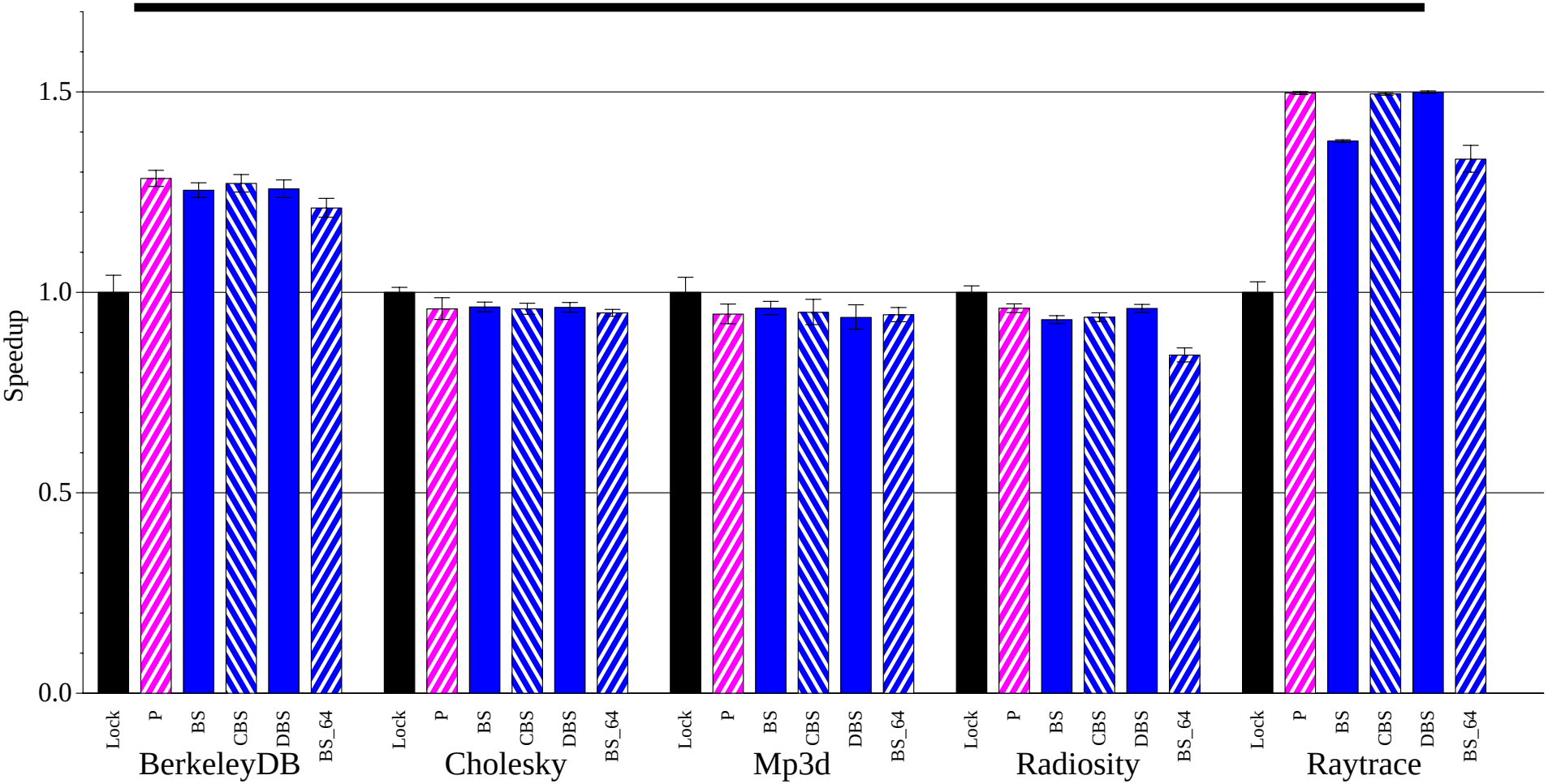


Perfect Signature Results



Perfect signatures similar or better than Locks

Realistic Signature Results



Realistic Signatures similar to Perfect Signatures and Locks

For our workloads, false positives are not a problem

What about scalability?

- Bigger system
- Bigger transactions

- False positives are a function of:
  - Transaction size
  - Transactional duty cycle
  - Number of concurrent transactional threads
  - Filtering due to on-chip directory protocol

- Signatures gracefully degrade to serialization

HTM Issues (Solutions)

- R/W bits in precious L1 cache design
  - R/W signatures: out of L1 cache & software-accessible
- Replicate R/W bits for SMT?
  - Easier to replicate signatures for SMT, but false positives
- Replicate again for (bounded) nesting?
- Save/restore R/W for thread switch?
- Modify for paging (virtual page moves)?

Outline

- Motivation
- LogTM Signature Edition (LogTM-SE) Hardware
- LogTM-SE in Operation
  - Unbounded Nested Transactions
  - Thread Switching
  - (Paging)
- Conclusions and future work

Unbounded Nesting Support

- Why?
  - Composability: libraries
  - Software Constructs: *Retry, OrElse* [Harris, PPoPP '05]
- What?
  - Signatures for each nesting level
- How?
  - One R / W signature set per SMT thread
  - Save / Restore signatures using Transaction Log

Nested Begin

Program

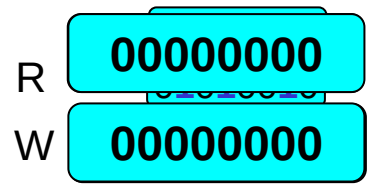
xbegin

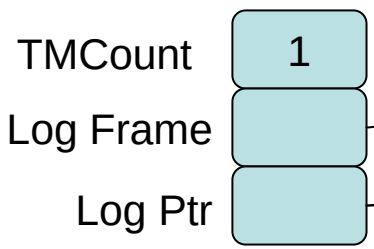
LD …

ST …

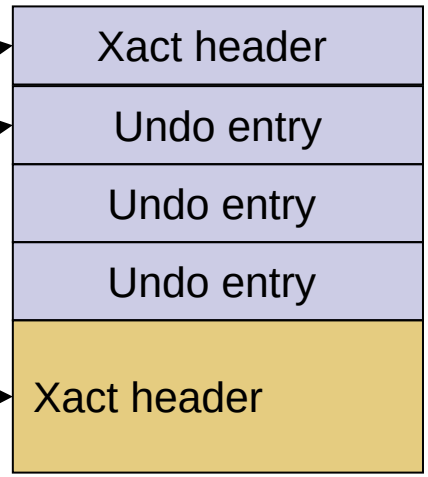
xbegin

Processor State





Transaction Log



Nested Begin

Program

xbegin

LD …

ST …

xbegin

Processor State

R 01001000

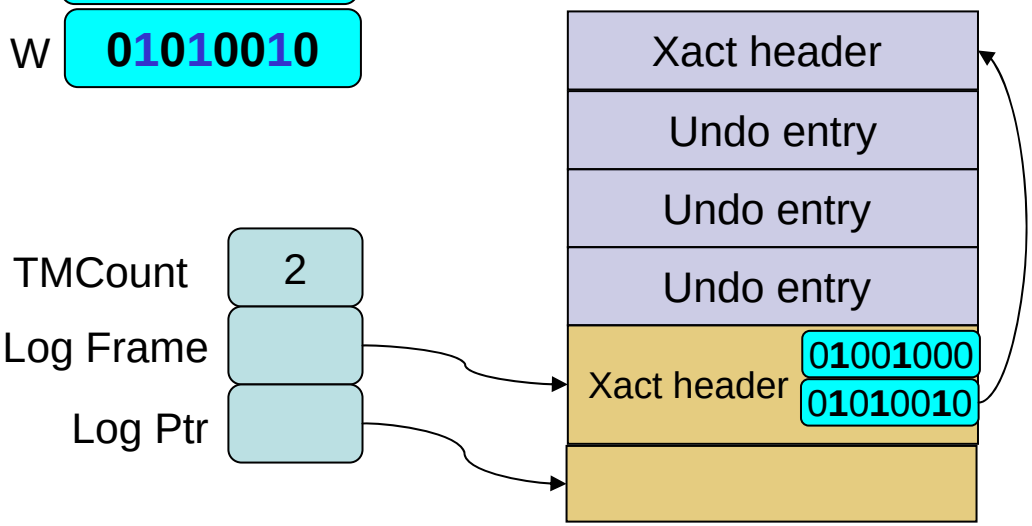
W 01010010

TMCount 2

Log Frame

Log Ptr

Transaction Log



Partial Abort

Program

xbegin

LD …

ST …

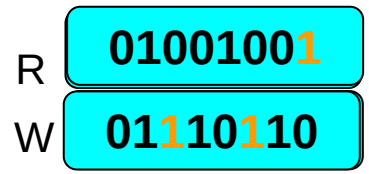
xbegin

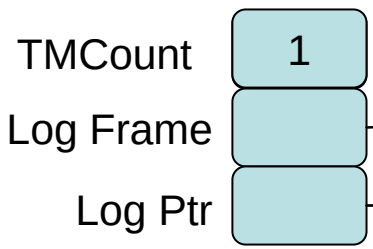
LD …

ST …

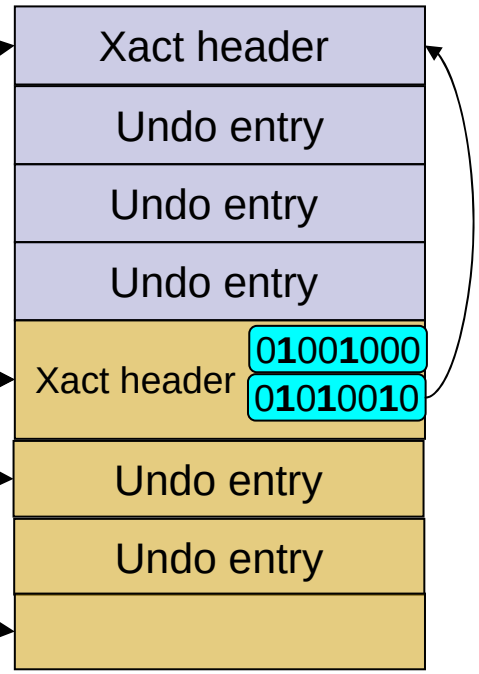
ABORT!

Processor State





Transaction Log



Nested Commit

Program

xbegin

LD …

ST …

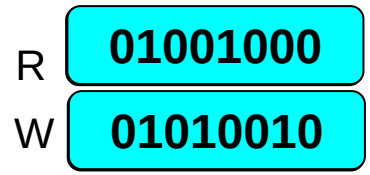
xbegin

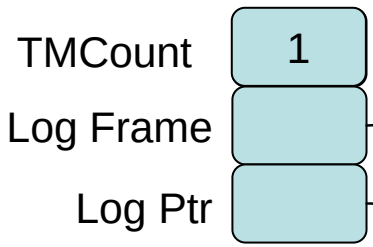
LD …

ST …

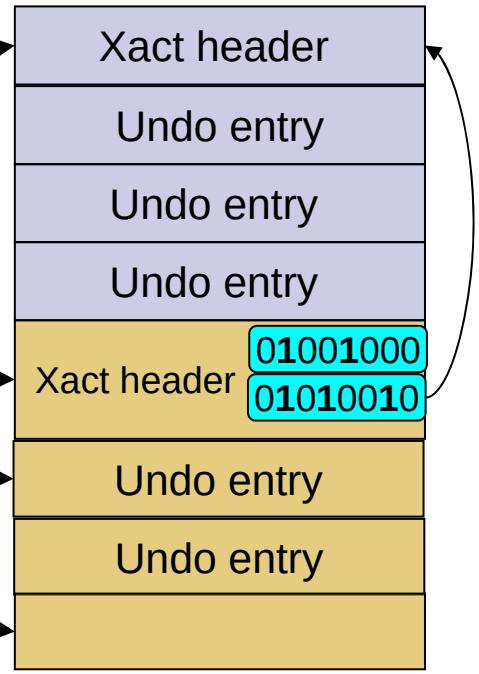
xend

Processor State





Transaction Log



Unbounded Nesting Support Summary

- Closed nesting:
  - Begin: save signatures
  - Abort: restore signatures
  - Commit: No signature action
- Open nesting:
  - Begin: save signatures
  - Abort: restore signatures
  - Commit: restore signatures

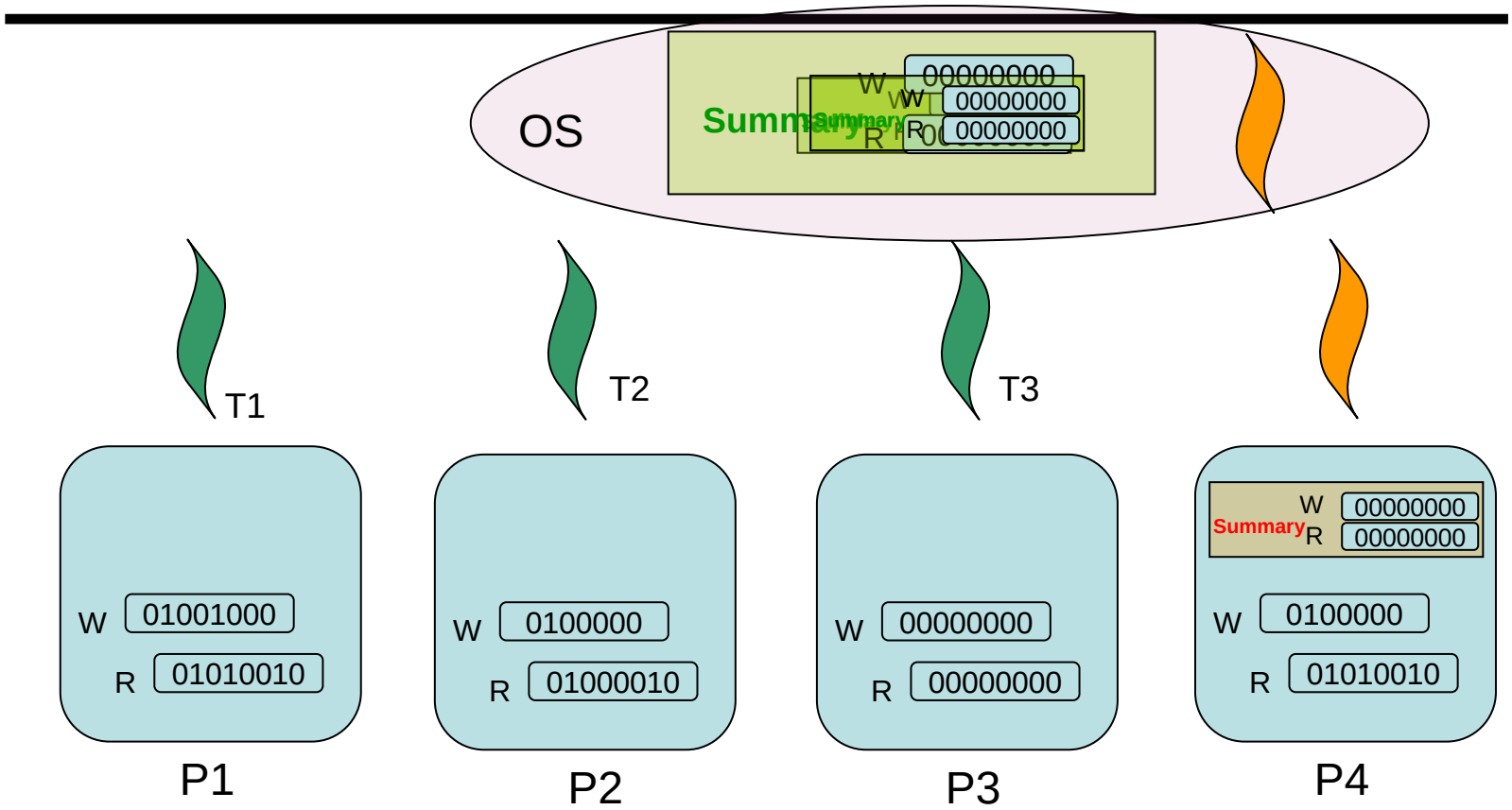
Thread Switching Support

- Why?
  - Support long-running transactions
- What?
  - Conflict Detection for descheduled transactions
- How?
  - ***Summary*** Read / Write signatures:

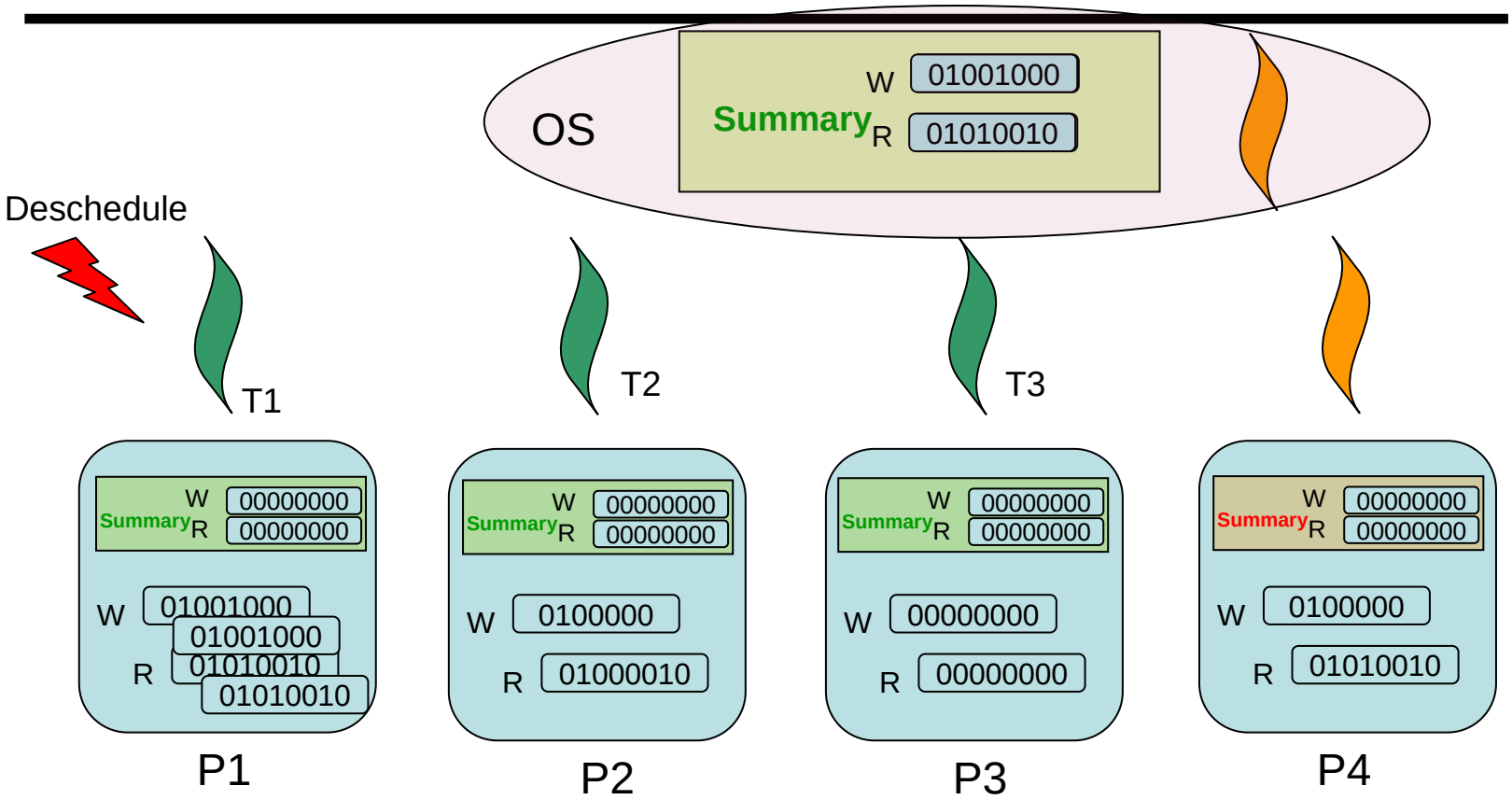
    If thread t of process P is scheduled to use an active signature, the corresponding summary signature holds the union of the saved signatures from all descheduled threads from process P.

    Updated using TLB-shootdown-like mechanism

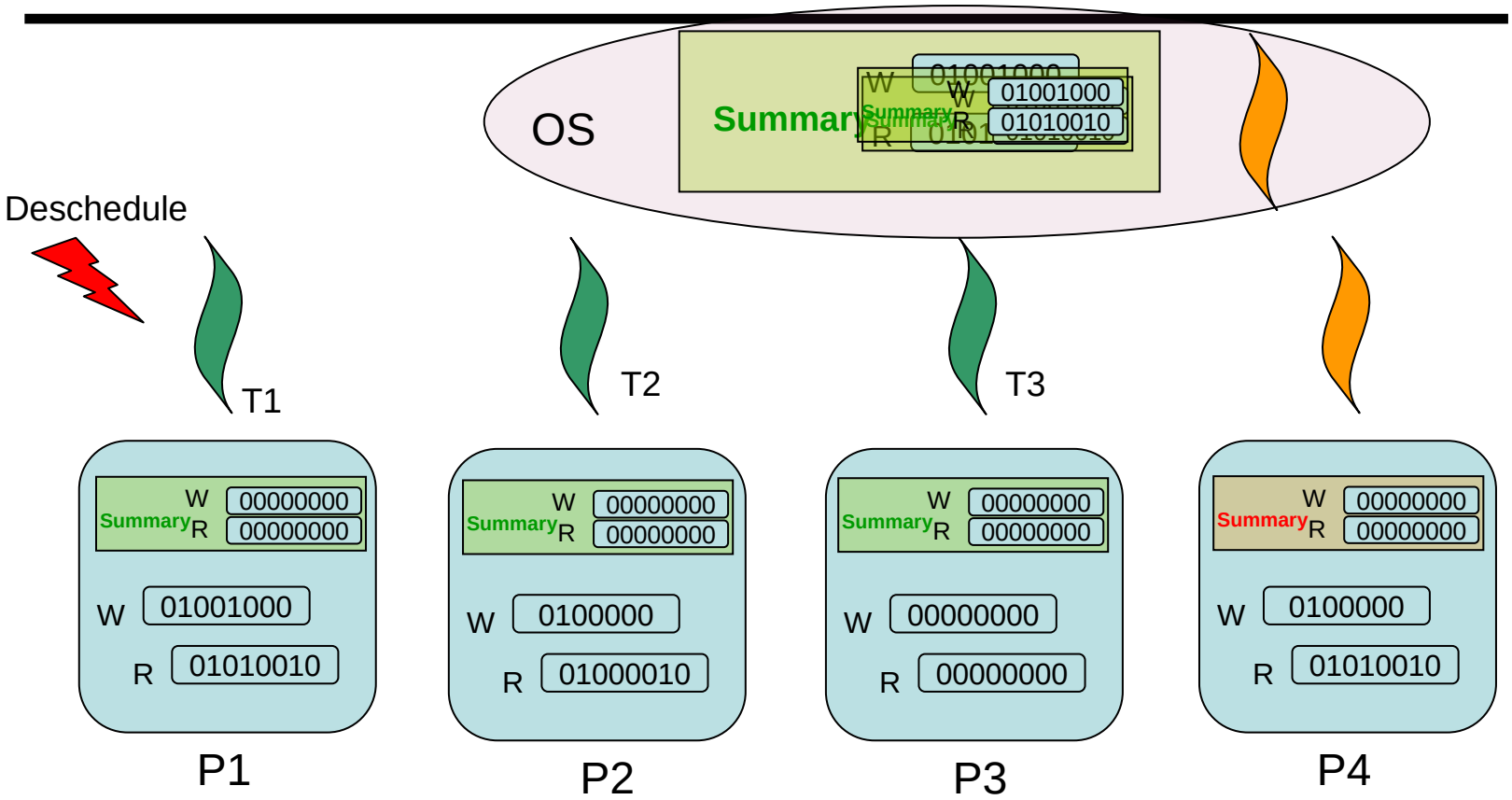
Handling Thread Switching



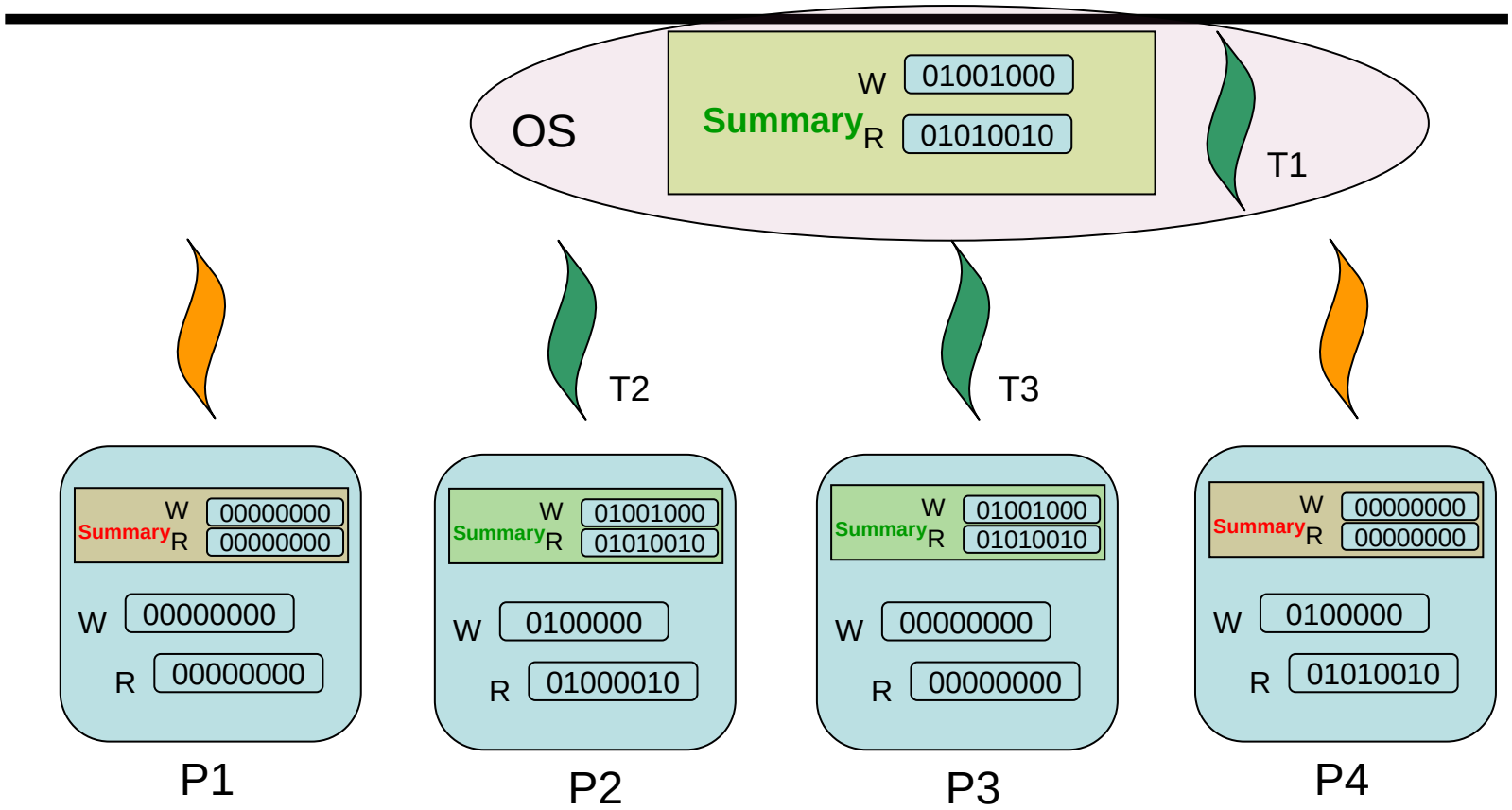
Handling Thread Switching



Handling Thread Switching



Handling Thread Switching



Thread Switching Support Summary

- ***Summary*** Read / Write signatures
  - Summarizes descheduled threads with active transactions
- One OS structure per process
- Check summary signature on every memory access
- Updated on transaction deschedule
  - Similar to TLB shootdown

Paging Support Summary

Problem:

- Changing page frames
- Need to maintain isolation on transactional blocks

Solution:

On Page-Out:

- Save Virtual -> Physical mapping

On Page-In:

- If different page frame, update signatures with physical address of transactional blocks in new page frame.

HTM Issues (Solutions)

- R/W signatures: out of L1 & software-accessible
- Replicate signatures for SMT, but false positives
- Replicate again for (bounded) nesting?
  - Signatures saved/restored on log
- Save/restore R/W for thread switch?
  - Signatures saved & distributed in summary signatures
- Modify for paging (virtual page moves)?
  - (Physical) signatures updated on page-in

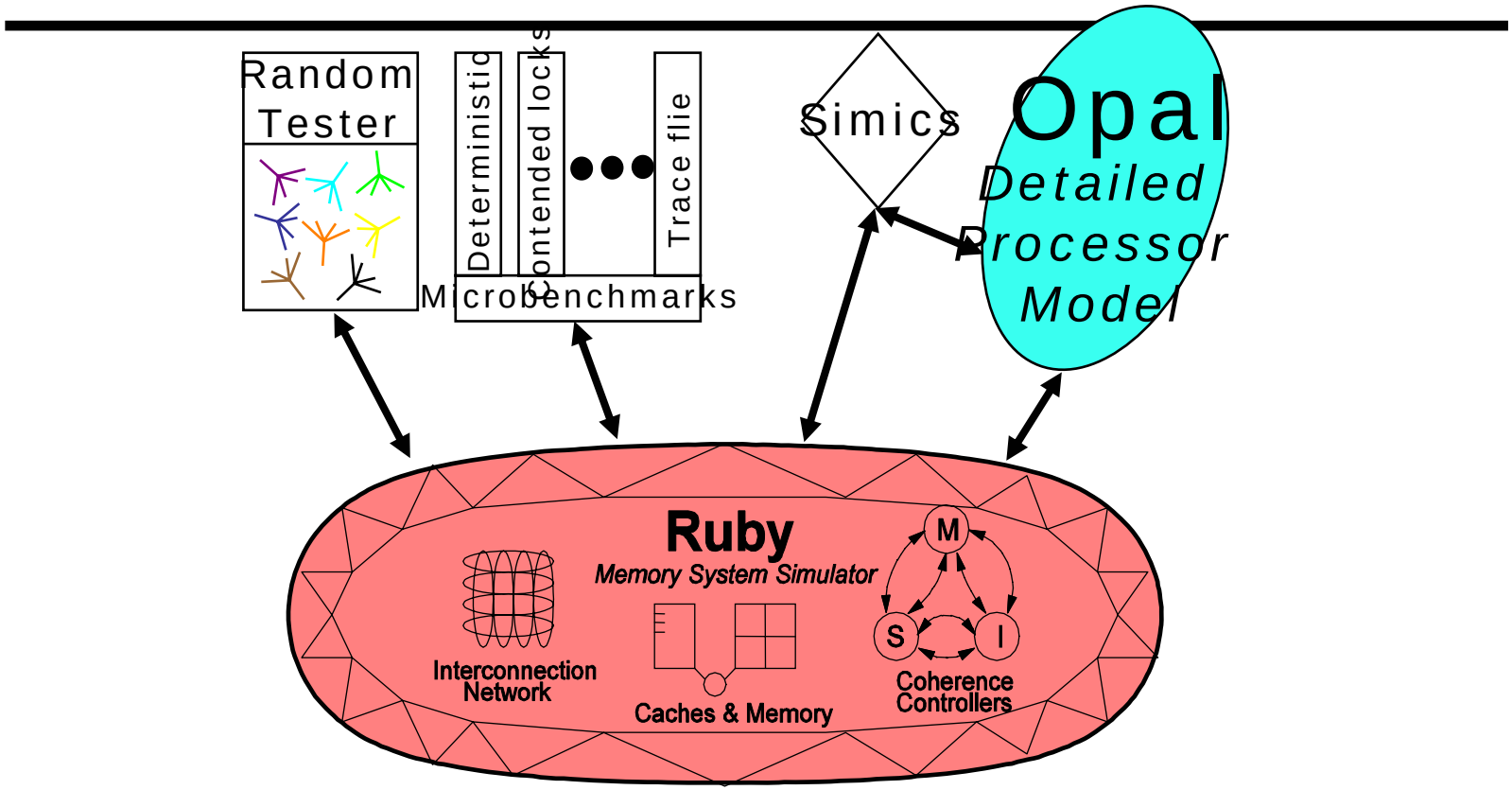
Future Work

- Explore scalability of signatures
  - Bigger system
  - Bigger transactions
- Modify OS for LogTM-SE
- Optimize OS-support for Reduced Overheads

Executive Summary

- Hardware Transactional Memory (HTM) Fast
  - HW handles old/new versions (e.g., write buffer)
  - HW handles conflict detection (R/W bits & coherence)
- But Closely Coupled to L1 cache
  - On critical paths & hard for SW to save/restore
- LogTM Signature Edition (LogTM-SE)
  - HW: LogTM's Log + Signatures (from Illinois Bulk)
  - SW: Unbounded nesting, thread switching, & paging
- Our Approach: Decoupled, Simple HW, SW control

Google “Wisconsin GEMS”



Works w/ Simics (free to academics) ➔ commercial OS/apps

SPARC out-of-order, x86 in-order, CMPs, SMPs, & LogTM

GPL release of GEMS used in four HPCA 2007 papers

GEMS News

- Version 1.4 available Friday:
  - LogTM bugfixes
  - MESI Single-CMP protocol
  - Fixes for GCC 4.X Compilation

  See **www.cs.wisc.edu/gems**

- GEMS 2.0: A future major LogTM release will include:
  - Single-CMP TM protocol
  - Signature support
  - Software abort handler

Backup

HTM Virtualization Mechanisms

	Before Virtualization				After Virtualization					
	$Miss	Commit	Abort	$Eviction	$Miss	Commit	Abort	$Eviction	Paging	Thread Switch
UTM	-	-	-	H	H	H	HC	H	H	H
VTM	-	-	-	S	S	SC	S	S	S	SWV
UnrestrictedTM	-	-	-	A	B	B	B	B	AS	AS
XTM	-	-	-	ASC	-	SCV	S	SC	SC	AS
XTM-g	-	-	-	SC	-	SCV	S	SC	SC	AS
PTM-Copy	-	-	-	SC	S	S	SC	SC	S	S
PTM-Select	-	-	-	S	H	S	S	S	S	S
LogTM-SE	-	-	SC	-	-	S	SC	-	S	S

Shaded = virtualization event
- = handled in simple HW
H = complex hardware

S = handled in software
A = abort transaction
C = copy values

W = walk cache
V = validate read set
B = block other transactions

LogTM Overview [HPCA '06]

- Version Management
  - Keep **old** values for abort **AND new** values for commit
  - **LogTM**:
    - **Eager**: record old values "elsewhere"; update "in place"
  - Other HTMs:
    - **Lazy**: update "elsewhere"; keep old values "in place"

← Fast Commit

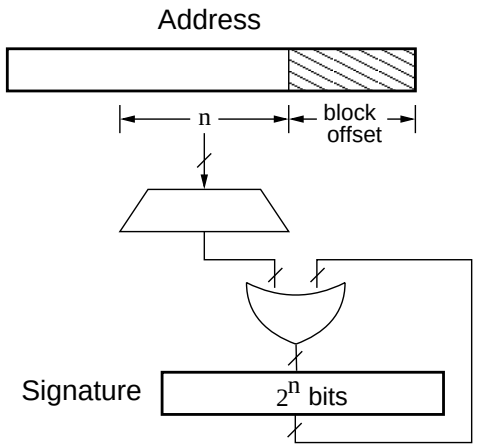
- Conflict Detection
  - Find read-write, write-read or write-write conflicts among concurrent transactions
  - **LogTM**:
    - **Eager**: detect conflict on every read/write
  - Other HTMs:
    - **Lazy**: detect conflict at end (commit/abort)

← Allows Stalling

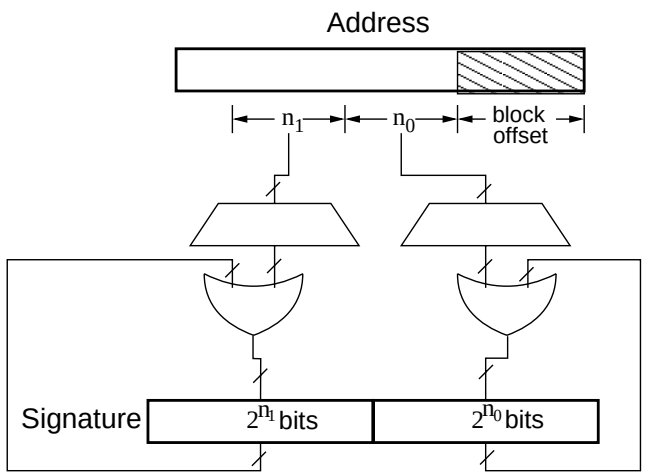
Benchmark Details

Benchmark	Input	Unit of Work	Units Measured
BerkeleyDB	1000 words	1 database read	128
Cholesky	Tk14.O	Factorization	1
Radiosity	batch	1 task	64
Raytrace	teapot	Whole parallel phase	1
Mp3d	128 molecules	1 step	1024

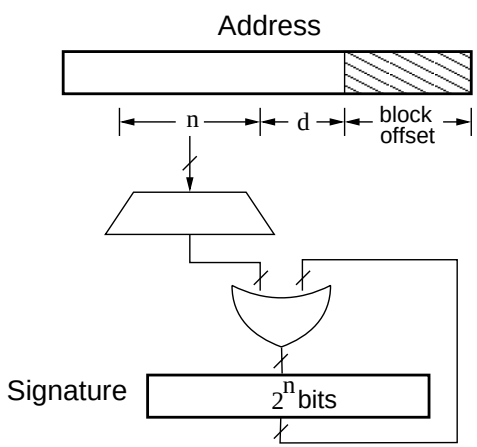
Signature Implementations



(a) bit−select (BS)



(b) double−bit−select (DBS)



(c) coarse−bit−select (CBS)

Results assume 2048-bit signatures, and operates on physical addresses

Paging Support Summary

- At Page Out,
  - Remember VP->PP

- At Page In,

if (VP->PP')

for each thread *t* in process

if *t* is in transaction:

Update signatures of t