

Log-Based Transactional Memory

Kevin E. Moore

Motivation

- Chip-multiprocessors/Multi-core/Many-core are here
 - “Intel has 10 projects in the works that contain four or more computing cores per chip” -- Paul Otellini, Intel CEO, Fall '05
 - We must **effectively program** these systems
 - But programming with locks is challenging
 - “Blocking on a mutex is a surprisingly delicate dance”
- OpenSolaris, mutex.c

Locks are Hard

```
// WITH LOCKS
void move(T s, T d, Obj key){
    LOCK(s);
    LOCK(d);
    tmp = s.remove(key);
    d.insert(key, tmp);
    UNLOCK(d);
    UNLOCK(s);
}
```

Moreover

Coarse-grain locking limits concurrency

Fine-grain locking difficult

```
Thread 0
move(a, b, key1);

Thread 1
move(b, a, key2);
```

DEADLOCK!

Transactional Memory (TM)

- Programmer says
 - “I want this atomic”
 - TM system
 - “Makes it so”
- ```
void move(T s, T d, Obj key){
 atomic {
 tmp = s.remove(key);
 d.insert(key, tmp);
 }
}
```
- Software TM (STM) Implementations
    - Currently slower than locks
    - Always slower than hardware?
  - Hardware TM (HTM) Implementations
    - Leverage cache coherence & speculation
    - Fast
    - But hardware overheads and virtualization challenges

## Goals for Transactional Memory

- Efficient Implementation
  - Make the common case fast
  - Can't justify expensive HW (yet)
- Virtualizing TM
  - Don't limit programming model
  - Allow transactions of any size and duration

## Implementing TM

- Version Management
  - **new** values for commit
  - **old** values for abort
  - **Must keep both**
- Conflict Detection
  - Find **read-write**, **write-read** or **write-write** conflicts among concurrent transactions
  - **Allows multiple readers OR one writer**

Large state  
(must be precise)

Checked often  
(must be fast)

## LogTM: Log-Based Transactional Memory

- **Combined Hardware/Software Transactional Memory**
  - Conservative hardware conflict detection
  - Software version management (with some hardware support)
- **Eager Version Management**
  - Stores new values in place
  - Stores old values in user virtual memory (the [transaction log](#))
- **Eager Conflict Detection**
  - Detects transaction conflicts on each load and store

03/07/2008

Wisconsin Multifacet Project

## LogTM Publications

- [HPCA 2006] [LogTM: Log-based Transactional Memory](#)
- [ASPLOS 2006] [Supporting Nested Transactional Memory in LogTM](#)
- [HPCA 2007] [LogTM-SE: Decoupling Hardware Transactional Memory from Caches](#)
- [ISCA 2007] [Performance Pathologies in Hardware Transactional Memory](#)

03/07/2008

Wisconsin Multifacet Project

## Outline

- Introduction
- Background
- LogTM
- Implementing LogTM
- Evaluation
- Extending LogTM
- Related Work
- Conclusion

03/07/2008

Wisconsin Multifacet Project

## LOGTM

University of Wisconsin-Madison

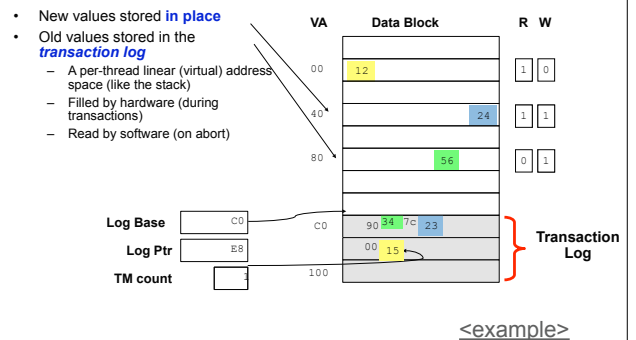
## LogTM: Log-Based Transactional Memory

- **Eager Software-Based Version Management**
  - Store new values in place
  - Store old values in the [transaction log](#)
  - Undo failed transactions in software
- **Eager All-Hardware Conflict Detection**
  - Isolate new values
  - Fast conflict detection for all transactions

03/07/2008

Wisconsin Multifacet Project

## LogTM's Eager Version Management



03/07/2008

Wisconsin Multifacet Project

## Eager Version Management Discussion

- Advantages:
  - No extra indirection (unlike STM)
  - Fast Commits
    - No copying
    - Common case
- Disadvantages
  - Slow/Complex Aborts
    - Undo aborting transaction
  - Relies on Eager Conflict Detection/Prevention

03/07/2008

Wisconsin Multifacet Project

## LogTM's Eager Conflict Detection

### Requirements for Conflict Detection in LogTM:

1. Transactions Must Be Well Formed
  - Each thread must obtain read isolation on all memory locations read and write isolation on all locations written
2. Isolation Must be Strict Two-Phase
  - Any thread that acquires read or write isolation on a memory location in a transaction must maintain that isolation until the end of the transaction
3. Isolation Must Be Released at the End of a Transaction
  - Because conflicts may prevent transactions from making progress, a thread completing a transaction must release isolation when it aborts or commits a transaction

03/07/2008

Wisconsin Multifacet Project

## LogTM's Conflict Detection in Practice

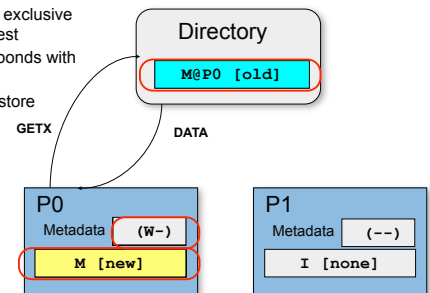
- LogTM **detects** conflicts using **coherence**
  - Requesting processor issues coherence request to memory system
  - Coherence mechanism forwards to other processor(s)
  - Responding processor **detects** conflict using local state & **informs** requesting processor of conflict
- Requesting processor **resolves** conflict (discussed later)

03/07/2008

Wisconsin Multifacet Project

## Example Implementation (LogTM-Dir)

- P0 store
  - P0 sends get exclusive (GETX) request
  - Directory responds with data (old)
  - P0 executes store

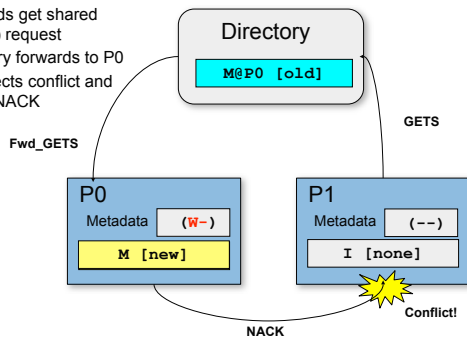


03/07/2008

Wisconsin Multifacet Project

## Example Implementation (LogTM-Dir)

- In-cache transaction conflict
  - P1 sends get shared (GETS) request
  - Directory forwards to P0
  - P1 detects conflict and sends NACK



03/07/2008

Wisconsin Multifacet Project

## Conflict Resolution

- Conflict Resolution
  - Can **wait** risking deadlock
  - Can **abort** risking livelock
  - **Wait/abort** transaction at **requesting** or **responding** proc?
- LogTM resolves conflicts **at requesting processor**
  - Requesting can **wait** (using coherence nacks/retries)
  - But must **abort** if deadlock is possible
- **Requester Stalls Policy**
  - Logically order transactions with timestamps
  - On conflict notification, wait unless already causing an older transaction to wait

03/07/2008

Wisconsin Multifacet Project

## LogTM API

| User                 | System/Library                           | Low-Level                     |
|----------------------|------------------------------------------|-------------------------------|
| begin_transaction()  | Initialize_logtm_transactions()          | Undo_log_entry()              |
| commit_transaction() | Register_abort_handler(void (*) handler) | Complete_abort_with_restart() |
| abort_transaction()  |                                          | Complete_abort_wo_restart()   |

03/07/2008

Wisconsin Multifacet Project

## IMPLEMENTING LOGTM

University of Wisconsin-Madison

## Version Management Trade-offs

- Hardware vs. Software Register Checkpointing
- Implicit vs. Explicit Logging
- Buffered vs. Direct Logging
- Logging Granularity
- Logging Location

03/07/2008

Wisconsin Multifacet Project

## Compiler-Supported Software Logging

- Software Register Checkpointing
  - Compiler generates instructions to save registers to transaction log
- Software-only logging
  - Compiler generates instructions to save old values and to the transaction log
- Lowest implementation cost
  - All-software version management
- High overhead
  - Slow to start transactions (save registers)
  - Slow writes (extra load & instructions)

03/07/2008

Wisconsin Multifacet Project

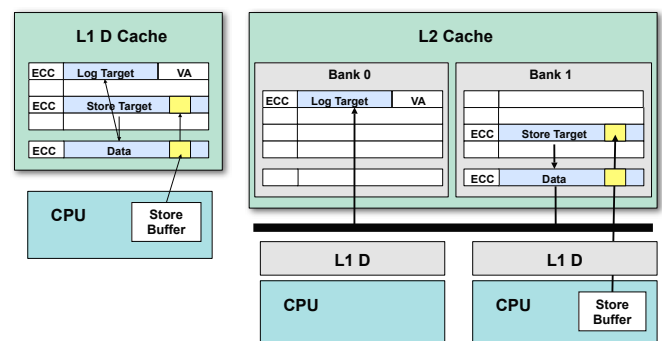
## In-Cache Hardware Logging

- Hardware Register Checkpointing
  - Bulk save architectural registers (like USIII)
- Hardware Logging
  - Hardware saves old values and virtual address to memory at the first level of writeback cache
- Best Performance
  - Little or no logging-induced delay
  - Single-cycle transaction begin/commit
- Complex implementation
  - Shadow register file
  - Buffering and forwarding logic in caches

03/07/2008

Wisconsin Multifacet Project

## In-Cache Hardware Logging



03/07/2008

Wisconsin Multifacet Project

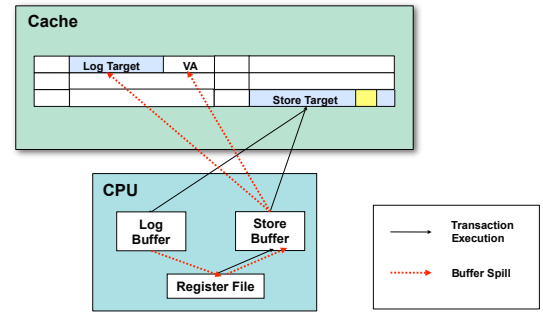
## Hardware/Software Hybrid Buffered Logging

- Hardware Register Checkpointing
  - Bulk save architectural registers (like USIII)
- Buffered Logging
  - Hardware saves old values and virtual address to a small buffer
- Good Performance
  - Little or no logging-induced delay for small transactions
  - Single-cycle transaction begin/commit
  - Reduces processor-to-cache memory traffic
- Less-complex implementation
  - Shadow register file
  - No changes to caches

03/07/2008

Wisconsin Multifacet Project

## Hardware/Software Hybrid Buffered Logging



03/07/2008

Wisconsin Multifacet Project

## Implementing Conflict Detection

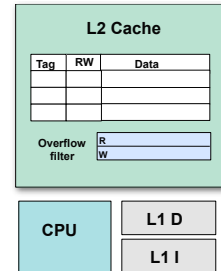
- Existing cache coherence mechanisms can support conflict detection for **cached data** by adding an R (read) W (write) bit to each cache line
- Challenges for detecting conflicts on **un-cached data** differ for broadcast and directory systems
- **Broadcast**
  - Easy to find all possible conflicts
  - Hard to filter false conflicts
- **Directory**
  - Hard to find all possible conflicts
  - Easy to filter false conflicts

03/07/2008

Wisconsin Multifacet Project

## LogTM-Bcast

- Adds a **Bloom Filter** to track memory blocks touched in a transaction, then evicted from the cache
- Allows any number of addresses to be added to the filter
- Detects all true conflicts
- Allows some **false conflicts**



03/07/2008

Wisconsin Multifacet Project

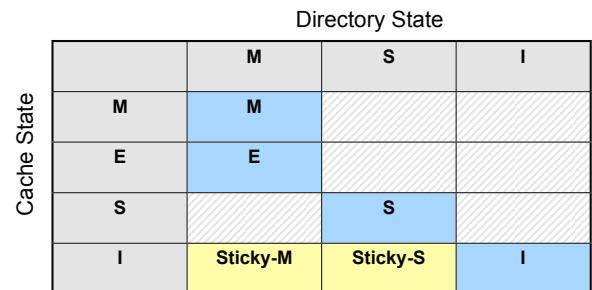
## LogTM-Dir

- Extends a standard MESI directory with **sticky states**
- The directory continues to forward coherence traffic for a memory location to processors that touch that location in a transaction then evict it from the cache
- Removes most false conflicts with a single **overflow bit** per cache

03/07/2008

Wisconsin Multifacet Project

## Sticky States



03/07/2008

Wisconsin Multifacet Project

### LogTM-Dir Conflict Detection w/ Cache Overflow

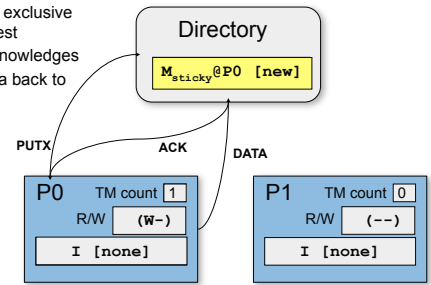
- At **overflow** at processor P0
  - Set P0's overflow bit (1 bit per processor)
  - Allow writeback, but set directory state to **Sticky@P0**
- At (potential) **conflicting request** by processor P1
  - Directory forwards P1's request to P0.
  - P0 tells P1 "no conflict" if overflow is reset
  - But asserts conflict if set (w/ small chance of false positive)
- At **transaction end (commit or abort)** at processor P0
  - Reset P0's overflow bit
- Clean sticky states **lazily** on next access

03/07/2008

Wisconsin Multifacet Project

### LogTM-Dir

- Cache overflow
  - P0 sends put exclusive (PUTX) request
  - Directory acknowledges
  - P0 writes data back to memory

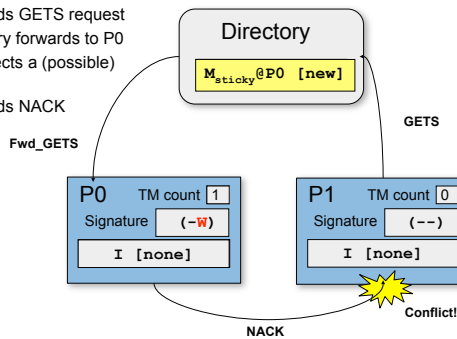


03/07/2008

Wisconsin Multifacet Project

### LogTM-Dir

- Out-of-cache conflict
  - P1 sends GETS request
  - Directory forwards to P0
  - P0 detects a (possible) conflict
  - P0 sends NACK

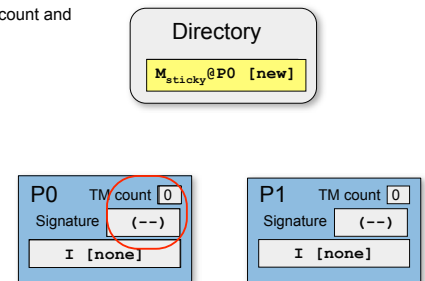


03/07/2008

Wisconsin Multifacet Project

### LogTM-Dir

- Commit
  - P0 clears TM count and Signature

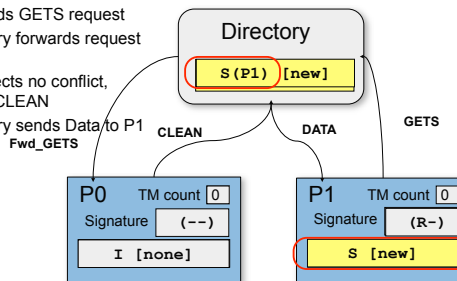


03/07/2008

Wisconsin Multifacet Project

### LogTM-Dir

- Lazy cleanup
  - P1 sends GETS request
  - Directory forwards request to P0
  - P0 detects no conflict, sends CLEAN
  - Directory sends Data to P1



03/07/2008

Wisconsin Multifacet Project

## EVALUATION

## System Model

- LogTM-Dir
- In-Cache Hardware Logging & Hybrid Buffered Logging

| Component               | Settings                                                       |
|-------------------------|----------------------------------------------------------------|
| Processors              | 32, 1 GHz, single-issue, in-order, non-memory IPC=1            |
| L1 Cache                | 16 kB 4-way split, 1-cycle latency                             |
| L2 Cache                | 4 MB 4-way unified, 12-cycle latency                           |
| Memory                  | 4 GB, 80-cycle latency                                         |
| Directory               | Full-bit-vector sharers list, directory cache, 6-cycle latency |
| Interconnection Network | Hierarchical switch topology, 14-cycle link latency            |

03/07/2008

Wisconsin Multifacet Project

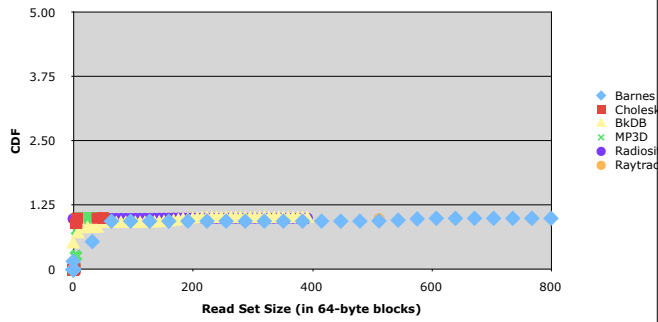
## Benchmarks

| Benchmark          | Synchronization             | Inputs                       |
|--------------------|-----------------------------|------------------------------|
| Shared Counter     | Counter lock                | 2500 cycle random think time |
| B-Tree             | Transactions only           | 9-ary tree, 5 levels deep    |
| Barnes             | Locks on tree nodes         | 512 bodies                   |
| Cholesky           | Task queue locks            | 14                           |
| Berkeley DB (BkDB) | Locks on object lists       | 512 operations               |
| MP3D               | Locks                       | 4096 molecules               |
| Radiosity          | Task queue locks            | Large room                   |
| Raytrace           | Work list and counter locks | Car                          |

03/07/2008

Wisconsin Multifacet Project

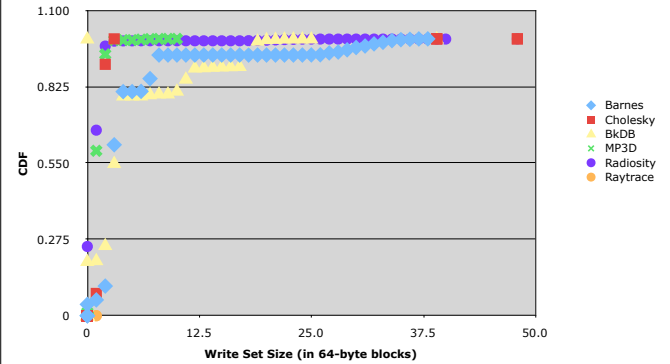
## Read Set Size



03/07/2008

Wisconsin Multifacet Project

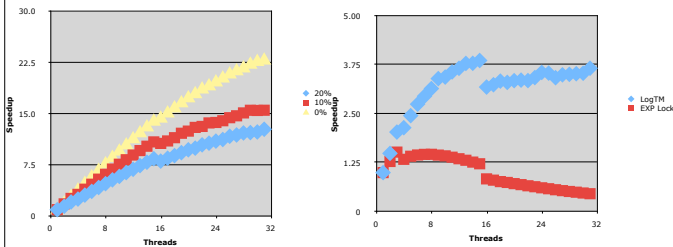
## Write Set Size



03/07/2008

Wisconsin Multifacet Project

## Microbenchmark Scalability



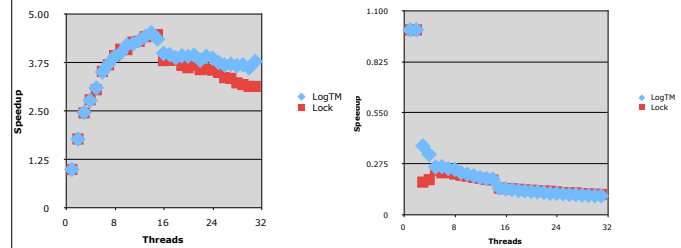
Btree 0%, 10% and 20% Updates

Shared Counter: LogTM vs. Locks

03/07/2008

Wisconsin Multifacet Project

## Benchmark Scalability

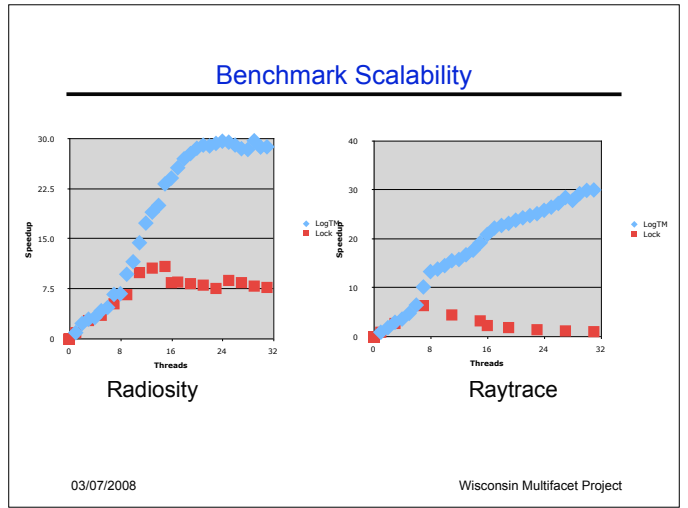
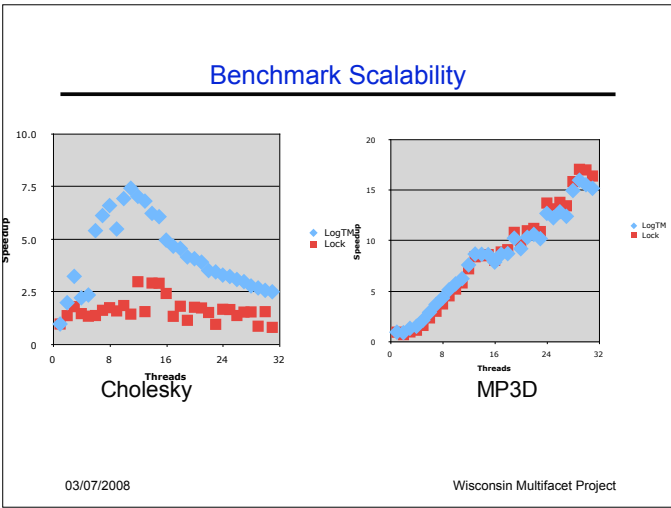


Barnes

BkDB

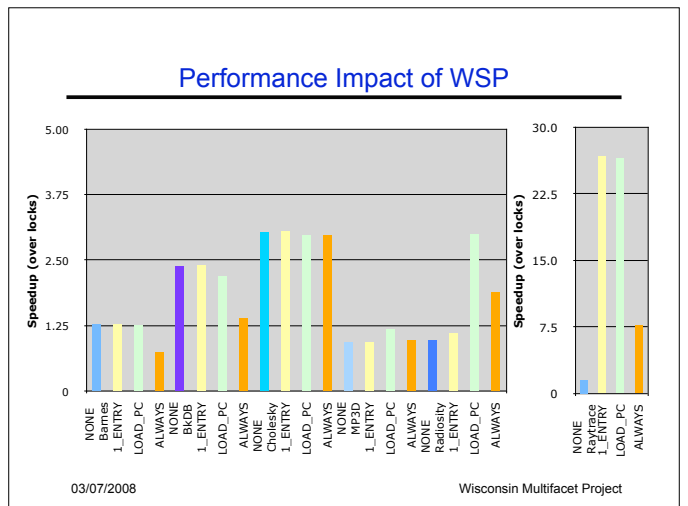
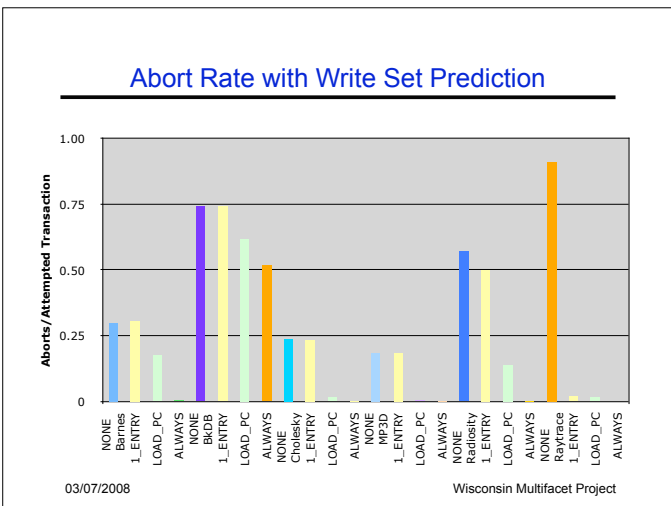
03/07/2008

Wisconsin Multifacet Project

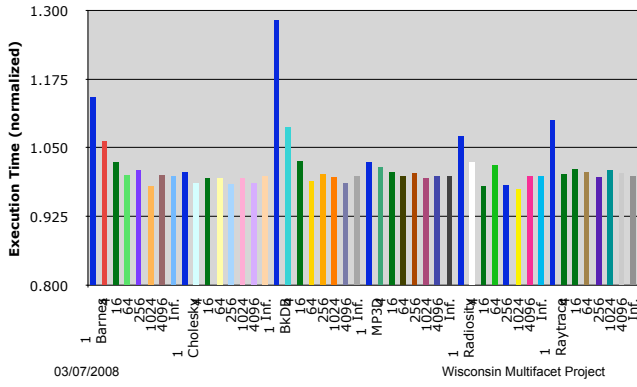


- ### Scalability Summary
- Benchmarks scale as well or better using LogTM transactions
    - Performance is better for all benchmarks
  - LogTM improves the scalability of some benchmarks, but not others
  - Abort rates are low
  - Next:
    - Write set prediction
    - Buffered Logging
    - Log Granularity
- 03/07/2008 Wisconsin Multifacet Project

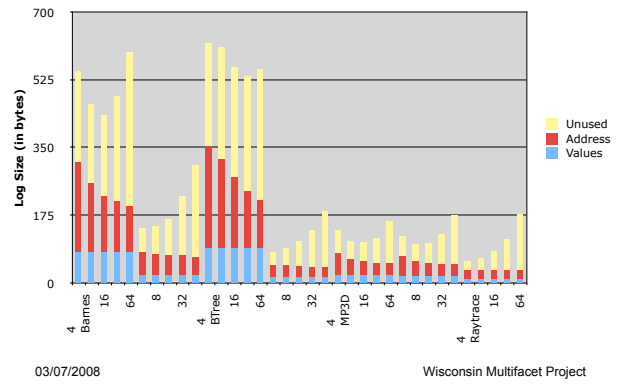
- ### Write Set Prediction
- Predicts if the target of each load will be modified in this transaction
  - Eagerly acquires write isolation
  - Reduces “waits for” cycles that force aborts in LogTM
  - Four Predictors:
    - **None** -- Never predict
    - **1-Entry** -- Remembers a single address
    - **Load PC** -- History based on PC of load instruction
    - **Always** -- Acquire write isolation for all loads and stores
- 03/07/2008 Wisconsin Multifacet Project



## Impact of Buffer-Spill Stalls



## Log Granularity



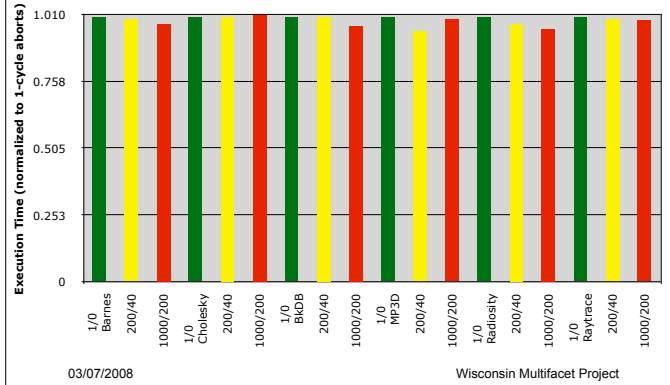
## Modeling Abort Penalty

- Abort penalty
  - Delays coherence requests
  - Delays transaction restart
- Penalty consists of:
  - Trap overhead (constant)
  - Rollback overhead (per log entry)
- Measured performance for 3 settings:
  - Ideal -- single-cycle abort
  - Medium -- 200 cycle trap, 40-cycle per undo record
  - Slow -- 1000 cycle trap, 200-cycle per undo record

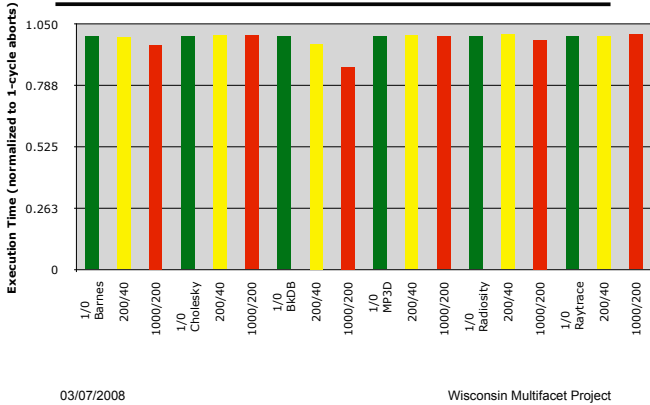
03/07/2008

Wisconsin Multifacet Project

## Sensitivity to Abort Penalty (no WSP)



## Sensitivity to Abort Penalty (with WSP)



## EXTENDING LOGTM

## Extending LogTM

- Supporting Nesting in LogTM
  - Support nested VM by segmenting the transaction log
  - Non-transactional escape actions facilitate OS interactions
- Virtualizing Conflict Detection with Signatures
  - LogTM-Signature Edition (LogTM-SE) tracks read and write sets with **signatures** (like Bloom Filters)
  - Supports thread switching and paging by saving, restoring and manipulating signatures

03/07/2008

Wisconsin Multifacet Project

## RELATED WORK

University of Wisconsin-Madison

## Related Work

- Hardware Support for Database Transactions
- Early Transactional Memory Systems
- Hardware TM (HTM)
- Software TM (STM)
- Hybrid TM
- TM Virtualization

03/07/2008

Wisconsin Multifacet Project

## Early Transactional Memory Systems

- Hardware Support for Database Transactions
  - **801 Storage System**
    - Database-like transactions on 1-level store (memory and disk)
    - Transactions are durable
- Early HTM
  - **Knight**
    - used transactions to parallelize code written in 'mostly functional' languages
  - **Herlihy and Moss**
    - First HTM
    - Implementation based on a separate transaction cache
    - Transactions limited to cached data

03/07/2008

Wisconsin Multifacet Project

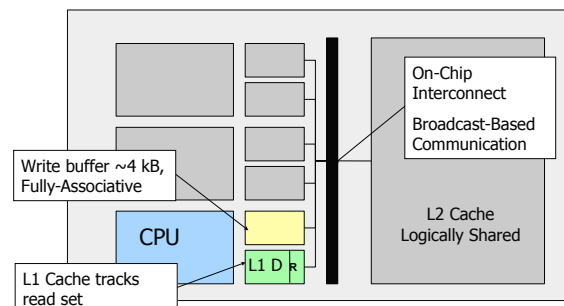
## Unbounded Transactional Memory

- Uses Eager VM and Eager CD
- Supports unbounded transactions in **hardware**
- Complex hardware
  - Pointer and state bits for each line in memory
  - Hardware state machine for transaction rollback
  - Global virtual address space

03/07/2008

Wisconsin Multifacet Project

## Transactional Memory Coherence and Consistency (TCC)



03/07/2008

Wisconsin Multifacet Project

## Bulk

- Encodes read and write sets in **signatures** (like bloom filters)
- Like TCC, uses lazy VM and lazy CD
- Can detect conflicts for non-cached data

03/07/2008

Wisconsin Multifacet Project

## Hybrid Transactional Memory

- Combines HTM and STM
- Executes small transactions in hardware, large transactions in software
- Allows program execution on existing hardware (without HTM support)

03/07/2008

Wisconsin Multifacet Project

## Transaction Virtualization

- Virtual Transactional Memory (VTM)
  - Rajwar and Herlihy
  - Adds a virtualization mechanism to limited HTM (e.g. Herlihy and Moss TM)
  - Implements CD and VM for transactions that exceed hardware capabilities in micro-code
- Page-granularity Transaction Virtualization
  - PTM -- Chuang et al.
  - XTM -- Chung et al.

03/07/2008

Wisconsin Multifacet Project

## HTM Virtualization Mechanisms

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

03/07/2008

Wisconsin Multifacet Project

## Conclusion

- TM can make parallel programs faster and easier to write
- LogTM provides:
  - Hardware/Software Implementation
    - Simple, flexible hardware
  - Software-Based Eager Version Management
    - Makes the common case (commit) fast
    - Reduces hardware complexity
  - Hardware-Based Eager Conflict Detection
    - Allows blocking to reduce wasted work

03/07/2008

Wisconsin Multifacet Project

## Thanks to my Collaborators

- Jayaram Bobba, Mark Hill, Derek Hower, Steve Jackson, Nick Kidd, Ben Liblit, Mike Marty, Michelle Moravan, Tom Reps, Mike Swift, Haris Volos, David Wood, Luke Yen

03/07/2008

Wisconsin Multifacet Project