Thread-Level Transactional Memory

Decoupling Interface and Implementation

UW Computer Architecture
Affiliates Conference
Kevin Moore
October 21, 2004
Transactional Memory

- Atomic and isolated execution
- Replace locks for critical sections
  - No lock granularity problem
- Software Error Recovery
  - Allow programmers to abort/rollback transactions when errors are detected
  - Convenient interface for exception handling
Why Haven’t You Built It?

- ISA Change
- No immediate benefit
- Complexity
- No solution for large transactions
- No solution for software error recovery
The Interface Matters

- Many systems implement the same ISA
  - Support multiple design points
  - Survive technology changes
- Programmability is the key motivation for transactional memory
- Interface challenge
  - Modern processors support small (few memory operations) easily
  - Unlimited transactions are more convenient for software
Virtualize Transactional Memory

- Decouple interface and implementation
  - Allow arbitrary size transactions
  - Don’t require unlimited buffering
- Flexible implementation
  - Multiple cost/performance options
- Support software error recovery
  - Recover all transactions
Overview

- Motivation
- Background
- TLTM Interface
- Out-of-Cache Transactions
- Conclusion
Prior Transactional Memory Systems

- Transactional Memory
- Transactional Memory Coherence and Consistency
- Oklahoma Update
- Transactional Lock Removal
- 801 Storage (Database Storage)
Transactional Memory

- Herlihy and Moss, ISCA 1993
- Replaced short critical sections with atomic transactions
- More efficient than locks
- Implementation based on cache coherence states
Herlihy and Moss, ISCA 1993

- **Transaction cache**
  - Stores all data accessed by transactions
  - 2 copies of each updated cache line
  - Fully associative
  - Acts as a victim cache

- **Long Locks**
  - Processors are allowed to refuse coherence requests
Transactional Memory
Coherence and Consistency

- TCC combines coherence and transaction support
- All memory requests are part of some transaction
- Designed for a single CMP
- Relies on broadcast
TCC

- Each processor executes a speculative (optimistic) transaction
- Commits are serial
  - Broadcasts addresses of all updates in the transaction
- Supports long-running (large) transactions
  - Must grab (and hold) commit permission first
  - Cannot abort large transactions
Thread-Level Transactional Memory

- **CPU**
  - **L1 D** Cache
  - **Write buffer ~4 kB, Fully-Associative**

- **L2 Cache** Logically Shared

- **On-Chip Interconnect**
  - **Broadcast-Based Communication**
Limits of Prior Proposals

- No separation between implementation and interface
- No concurrent execution of large transactions
- No software error recovery
- Don’t address operating system
Thread-Level Transactional Memory (TLTM)

- Decouple transactional interface and implementation
- No limit on transaction size or content
- Break the dependence on caching all data accessed in a transaction
TLTM API

- Transactions
  - Specified by *begin* and *commit* operations
  - All memory operations between begin and commit are part of the transaction
  - Transactions end in a *commit* or an *abort*

- Abort/Recovery
  - Can be triggered by software or hardware
  - Control may be passed to software abort handler
  - If software handler is called, a before-image log is provided in the memory space of the process
TLTM Guarantee

For each transaction the hardware will:

1) Execute the transaction atomically (and commit) or,

2) Allow a software abort handler to run before any updates are exposed
Transactional Memory Requirements

- Maintain multiple versions of updated objects
- Track data set (reads and updates)
- Rollback/Restart
Thread-Level Log

Thread 0
- Stack
- Log

Thread n
- Stack
- Log

Global
- Code
- Data
- Heap

October 21, 2004
Thread-Level Transactional Memory

18
Log-Based Recovery

- **Undo log**
  - Before image log stored in virtual address space of user program
  - Separate log for each thread
- Aborts can be handled in software
  - Replays log entries in LIFO order
- Breaks dependence on cache for 2nd copy
Speculative Transactions

- ULTM does not require logging
- Processors can still execute transactions speculatively
- Non-speculative transactions will be rare for many applications
Transaction Read Set Size

![Graph showing transaction read set size](image)

- **PDF**
- **BARNES 512**
- **OCEAN 66**

**Axes:**
- Y-axis: Probability density function (PDF)
- X-axis: Read set size (in 64-byte lines)
Transaction Write Set Size

![Graph showing PDF and Log Size for BARNES 512 and OCEAN 66]
Tracking Transaction State

- Use conservative estimate of transaction data sets
  - Allows bounded state
- Allows a trade-off between space overhead and performance
  - Results in some unnecessary aborts
Advantages of TLTM

- Flexibility for programmers
  - No hardware-imposed limits on transaction content
  - Allows conservative synchronization
  - Supports recovery for all transactions
- Flexibility for memory system designers
  - No minimum size or associativity requirement for caches, reorder buffers or store queues
  - Can off-load recovery to software