PPoPP '07

Alert-on-Update: A Communication Aid for Shared Memory Multiprocessors

Michael F. Spear, Arrvindh Shriraman, Hemayet Hossain,
Sandhya Dwarkadas, and Michael L. Scott
University of Rochester
www.cs.rochester.edu/research/synchronization/

The Key Idea

Simple HW mechanism to asynchronously notify program of eviction/invalidation of pre-specified cache lines. Used (in our experiments) to enable fast, non-blocking software transactional memory.

Hardware Mechanism

- Instructions to mark cache lines, establish/re-enable handler
- Spontaneous user-space subroutine call when marked line is invalidated, evicted or (optionally) written
- Implemented in private L1 or (with thread IDs) shared L2—one bit per line per context
- Alerts masked while in handler; delivered on re-enable
- Special registers hold address of line and type of alert (may say "lost while masked")
- Virtualization: clear AOU bits on context switch; invoke handler on reschedule: does full validation, remarks desired lines

SW Transactional Memory

- Alternative to locks
  - » system optimistically pursues concurrent atomic tasks
  - » rolls back and retries on conflict
- May be blocking or nonblocking
  - » NB SW typically uses indirection to install new object versions by swinging pointers

AOU for STM

- Mark object header for conflict detection
- Mark transaction descriptor for immediate aborts
- Keep estimate of cache capacity; scale back gradually on overflow
- Defer aborts for memory management, logging, etc.

The Validation Problem

- Must prevent erroneous behavior due to inconsistent reads in doomed transactions
- Major contributor to cost of correct STM
- Example:

```
object A {
    shared bool Bf_is_pointer;
}
object B {
    union { int n; int* p; } f;
}
T1: atomic {
    open A;
    bool b = A.Bf_is_pointer;
    open B;
    if (b) *B.f = 3;
}
```

- What happens if T2 commits changes to A and B here ?

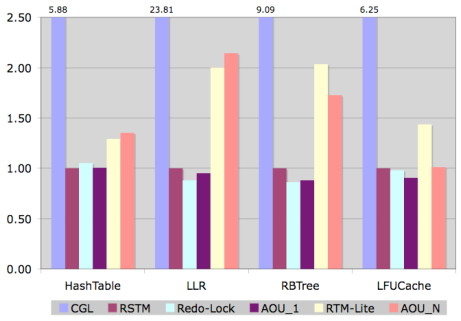
Alternative Solutions

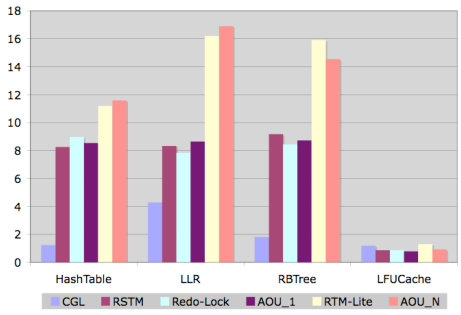
- Sandboxing
  - » needs language/compiler support; infeasible for C/C++
  - » Cf: HASTM [Saha et al., MICRO'06]
- Immutable clones, validate on open
  - » visible readers: validation cheap but visibility expensive
  - » invisible readers: validation quadratic **[RSTM]**
  - » AOU (or HASTM): visibility cheap, validation free **[RTM-Lite]**
- In-place update, validate object on access, validate all on open
  - » even if cheap, per-read object validation adds up
  - » during log application, must block **[Redo-Lock]** or steal **[AOU-1]**
    - nonblocking implementation easy with AOU
    - very complex without; see Marathe & Moir poster
  - » For bounded transactions, AOU also enables elision of per-object validation, using immediate aborts **[AOU-N]**
- Coarse-grain locks **[CGL]**—basis for compoarison

Other Uses of AOU

- Active messages
- Fast mutex: thread switch on failed acquire; alert on availability
- Fall rollback in NB algs
- ABA avoidance
- Debugger watchpoints
- Code security: buffer overflow protection (Cf. DieHard), read-only fields
- Software transactional memory (STM—the focus here)

Simics/GEMS evaluation:
16-way CMP; 1.2GHz in-order single-issue processor; AOU in 64KB 4-way split L1 w/ 64-byte blocks, 1 cycle latency, & 32-entry VB; 8MB 8-way unified L2 w/ 64-byte blocks & 20 cycle latency





For Further Information

- "Hardware Acceleration of Software Transactional Memory." Shriraman, Marathe, Dwarkadas, Scott, Eisenstat, Heriot, Scherer, & Spear. URCS TR 887, Dec.'05; TRANSACT'06.
- "Nonblocking Transactions Without Indirection Using Alert-on-Update." Spear, Shriraman, Dalessandro, Dwarkadas, & Scott. SPAA'07.
- "An Integrated Hardware-Software Approach to Flexible Transactional Memory." Shriraman, Spear, Hossain, Dwarkadas, & Scott. ISCA'07