

# Intel® Transactional Synchronization Extensions (Intel® TSX) Linux update

Andi Kleen  
Intel OTC

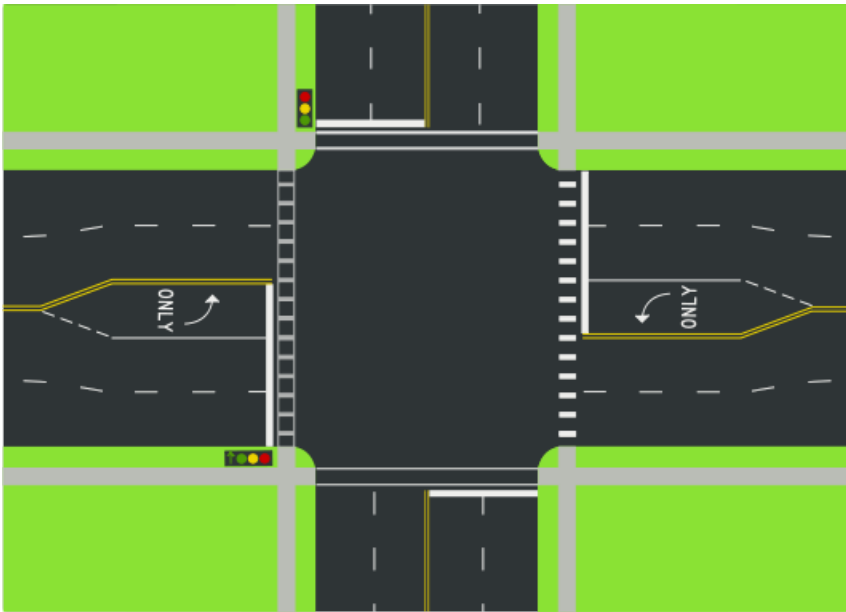
Linux Plumbers  
Sep 2013

# Elision

- Elision : the act or an instance of omitting something : omission

# On blocking

Blocking



Non-blocking

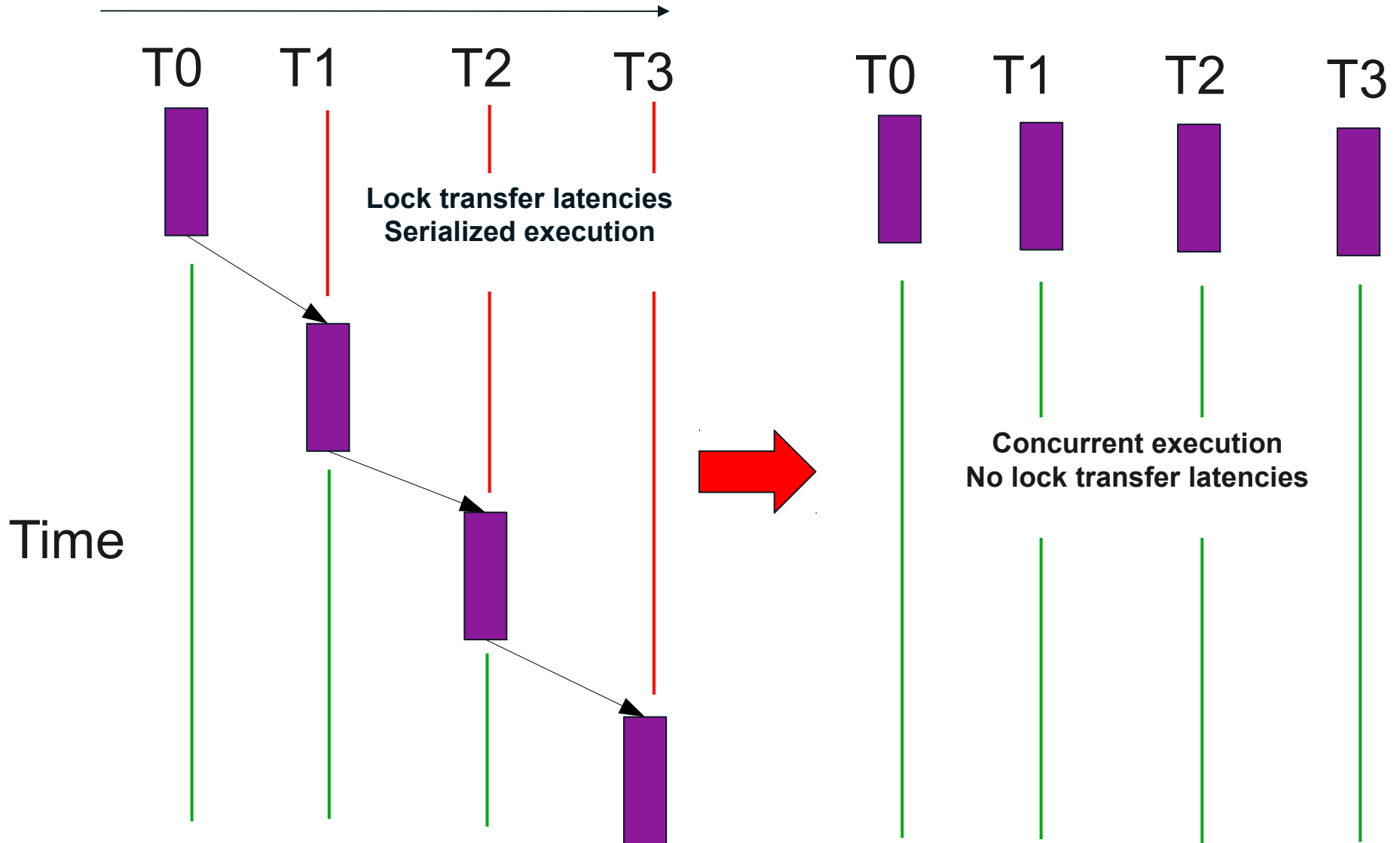


Credit: Wikipedia/Mintguy/Frederik/Romanm  
Creative Commons Attribution share alike 3.0

Credit: Wikipedia/Mintguy/Frederik/Romanm  
Creative Commons Attribution share alike 3.0

Idea from Dave Boucher

# Speculative execution



# Intel TSX quick overview

**User controlled speculative transactional execution mode in CPU**

**Implemented in Intel© 4th generation Core® (“Haswell”)**

Two ISA interfaces (HLE/RTM) to specify transactions

Transactions best effort (need fallback path)

**HLE adds XACQUIRE/XRELEASE instruction prefixes for atomic ops**

Lock elision done transparently in same code path

Nop on old CPUs

**RTM uses new XBEGIN/XEND instructions**

Explicit abort handler, can use lock as fallback path

**XTEST and XABORT**

# Basic RTM elided lock

```
elided_lock(lock) {  
    if (_xbegin() ==  
        _XBEGIN_STARTED) {  
        if (lock is free)  
            // in read set  
            return;  
        _xabort(0xff);  
        //0xff is lock busy  
    }  
    // come here on abort  
    original locking code  
}
```

```
elided_unlock(lock) {  
    if (lock is free)  
        _xend();  
    else  
        unlocking code  
}
```

- Simple wrapping code pattern
- Original lock code

# An elided lock ...

- Is a fast path
- Is non-blocking
- Acts mostly like a recursive reader lock
- Locks every cache line individually
- May always fall back
- **Uses the standard locking programming model**

# Linux Implementation overview

- Kernel TSX perf profiling support
  - Needed to understand speculation
- Kernel TSX lock elision
  - Elide kernel locks
- Glibc mutex elision
  - Elide application locks
- Various custom locks elided in applications
- Libitm (gcc)
- Applications with non scalable locking primary target



# Lock adaptation

- RTM locks with adaptive abort handler
  - Skips elision when not successful
- Safety net to avoid regressions
  - And prevents us from having to exercise all possible locks/workloads
- Simple algorithm used currently
  - State stored in lock itself
  - Lots of tunables, still looking for best configuration
  - Area for future work

# Future user elision work

- Extend glibc code: recursive locks, rwlock
- Better adaptation algorithms
- Tuning interface
- Go beyond POSIX
  - C++11 locking, new lock interface for C
  - Need new interface to elide condition variables
  - Will also allow better fast path without dynamic dispatch
  - Make adaptive spinning the default

# Enabling applications

- Applications with their own lock library
- RTM wrapper or HLE
- Challenge: sometimes custom locking spread over the code
- Need to identify critical sections
- After enabling typically some tuning is useful to lower aborts
  - For example avoid statistic counters
  -

# Kernel elision

- Eliding mutex, spin, rw, bitspin, rwsem, custom lock
- Kernel is already quite scalable
- Need some changes to lower conflicts
  - Most of these changes help without elision
- Only win in some areas with big locks
  - Occasional losses due to too fine grained locking
  - May benefit from lock coarsening

# Kernel locks that do not elide well (without changes)

- MMIO, writing to DMAed data (device driver)
- TLB flush (write faults)
- IPI, APIC, MSR accesses
- Scheduler (high chance of conflict, WIP)
- BUG\_ON(!\*\_is\_locked())
- “meta locking” (btrfs)
  - But btrfs would benefit a lot!
- False sharing
- Common conflicts, (page allocator, ext4 extent trees, reference counts)
- Rewriting unchanged data often (flags)
- Large copies, clears: capacity (read,write)
- Unmatched lock\*\_irq/unlock

**Handled with adaptation, some minor annotation and changes**

# References

- <http://www.intel.com/software/tsx>
- Intel optimization guide, Chapter 12
- Glibc: <http://github.com/andikleen/glibc>
- Kernel:
  - <http://git.kernel.org/pub/scm/linux/kernel/git/ak/linux-misc.git>
  - hle\*/combined elision branches
  - hsw/pmu\* perf support
  - Branch names are changing!