Restricted Transactional Memory (RTM)

Part of the Intel Transactional Synchronization Extension

August 2016

Francis Giraldeau
francis.giraldeau@polymtl.ca

Prof. Michel Dagenais
michel.dagenais@polymtl.ca

Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL

ERICSSON

Critical section

- Lock prevents race condition
- Lock must be taken every time
- Usually, low probability of conflict
- Therefore, most of the time, the lock is taken for nothing, causing cache line bouncing and overhead
- Lock granularity:
  - Coarse granularity increase the chance of conflict
  - Fine granularity increases memory usage and cache line replication overhead

Memory transaction

- Speculative execution that aborts on conflict
- The transaction begins, checkpoint (XBEGIN)
- The processor snoops memory accessed by all cores
- Conflict with another core (XABORT)
  - Restore the state at checkpoint, throws changes
  - Jump to the beginning of the critical section
- No conflict: commit memory changes (XEND)
- No lock in the normal case!

Verify RTM hardware support

- Intel Haswell and beyond
- Complex circuit for high-end processors
- grep rtm /proc/cpuinfo

Example

```
#include <immintrin.h>


TransactionScope(SpinLock *lock) :m_lock(lock), m_code(0) {
    m_code = _xbegin();
    if (m_code == _XBEGIN_STARTED) {
        if (!m_lock->isLocked()) {
            return;
        } else {
            _xabort(1);
        }
    } else {
        m_lock->lock();
    }
}

~TransactionScope() {
    if (m_lock->isLocked()) {
        m_lock->unlock();
    } else {
        _xend();
    }
}
```

rtm instructions header

If this test is false, the previous transaction attempt failed

Lock-less critical section
Lock added to the read set

Fallback path (normal lock)

End of transaction (or locked section)