Using Intel® Transactional Synchronization Extensions from OpenMP® codes
Jim Cownie, Intel Corporation (UK) Ltd.

Background: What is Intel® TSX?

Transactional Memory:
• Speculative execution while collecting read and write sets
• On conflict (I read, someone else writes; I write, someone else writes) abort speculation
• On abort retry or fall back to locking (there are no progress guarantees in Intel® TSX itself)

Two ISA interfaces
• New prefixes on lock and unlock instructions (“HLE”). Binary compatible (ignored on older cores)
• New instructions (Restricted Transactional Memory “RTM”)

Why would I use it?
Because:
• You have critical sections where the lock is contended, but the data is not, e.g. a hash-table
• You have critical sections where the critical section is small and the cost of moving the lock cache-line is large
• It is simple to do in OpenMP, so worth trying
• It is available in modern Intel® CPUs

OpenMP 4.5 lock enhancements

Addition of a “hint” to lock initialization and critical sections
• Use omp_init_lock_with_hint instead of omp_init_lock
• No need to change other lock operations; you only need to touch lock initialization
• Hints: omp_lock_hint_*
  • contended/uncontended
  • speculative/nonspeculative
Hints are portable; they can always be ignored.
(In OpenMP 5.0 hint names will change to omp_sync_hint_* and can also be used on atomics)

Conclusions

Hashtable: OpenMP lock hints allow us to use Intel® TSX to produce scaling, thread-safe, data-structures (here a hashtable) without having to rewrite the system implementation to add fine-grained reader/writer locks

Atomic Vector: Atomic operations generally perform better than using Intel® TSX

Overall:
• It is easy to use Intel® TSX from OpenMP
• Where critical sections (rather than atomics) are required Intel® TSX can be beneficial