Transactional Memory

Preeti U Murthy
Energy Management Team
Linux Technology Center
February 24th,2013

What is Transactional Memory ?

- Synchronization Mechanism for SMP systems, which treats memory accesses like database transactions.
- The transaction occurring on a processor can either commit or abort towards its end.
- Commit occurs when no other processor has accessed the same data set, abort occurs otherwise.

The end effect of Transactional memory

- Transactional memory thus ensures that only one transaction can succeed, else no transaction succeeds. Thus assuring atomicity of a transaction.
- If a transaction aborts, then it needs to retry.

Quiz: If all transactions abort and retry, will the scenario (abort and retry) not repeat?

Deciding between Commit and Abort

- Cache coherence protocols are used to decide between commit and abort.
- If the data set of this transaction has been accessed in the local cache of another processor, abort all such transactions.

Difference between cache coherence protocol and TM

- Cache coherence protocols will invalidate the cache of other processors, if modified locally, **immediately**. This would lead to a cache miss on other processors the next time they would access it.
- However in TM, although a similar mechanism in which processors read into their local cache is followed, the invalidation is restricted to the **end of the transaction** if it has been written to elsewhere. This will minimize the overhead of cache misses.

Advantages of TM over Locking schemes

- Locking primitives bring with them their specific complexities like deadlocks, priority inversion, unpredictable locking duration. Transactional memory is free from these issues.

Advantages of TM over Locking schemes

- Usually locks are applied to data broken down to the finest level, so as to avoid heavy contention. Therefore large data structures are broken down into partitions, each of which becomes a lockable entity.

  But what happens if the data structure is non partitionable? Here is where transactional memory plays an important role.

Will the way we write programs change if TM is used?

- Yes. TM is applied to small transactions dealing with non partitionable data structures. Small transactions because, the overhead of retry can be minimized.
- These transactions need to be bracketed between some annotations to indicate to the hardware that this needs to be synchronized via TM. These annotations can be hidden in libraries.

Hardware support for TM

- The hardware already has the support to ensure cache coherence. So it will know if the transaction needs to be aborted or committed.
- However the additional support that needs to be added is *restore checkpoint*; meaning if the transaction aborts, the instruction pointer needs to be set back to the beginning of the transaction.
- All modified cache lines of the aborted transaction need to be discarded.
- If an 'x' number of re-tries fail, there must be a backup synchronization technique.

Weaknesses of TM

- In a highly conflicted data structure, the number of re-tries could be many.
- This could lead to certain nasty situations like, large transactions being starved by many smaller transactions due to many aborts.
- Or higher priority transactions being starved by many lower priority transactions.

Where is it best to use TM?

- In non-partitionable data structures where locking would lead to heavy contention due to many updatable entities in the data structure.
- When updates can be expected to touch minimal number of nodes simultaneously, thus minimizing conflicts, TM would be the obvious choice.
- Transactions ought to be small so as to minimize the number of retries and offset the advantages of TM.
- Whenever there is a possibility of a dead lock. like out of order acquiring of locks, better to use TM.
- Best applications of TM include Java/JVM and DB2.
- TM in kernel is yet to be evaluated.

References

- Is Parallel Programming Hard, And, If So, What Can You Do About It? By Paul McKenney.
- Why The Grass May Not Be Greener On The Other Side: A Comparison of Locking vs. Transactional Memory.
- The Case for Hardware Support for Transactional Memory by Christos Kozyrakis, Computer Systems Lab Stanford University.
- AskVaidy.com ;)