Proving Conflict Serializability for Full Invalidation

Workshop on the Theory of Transactional Memory (WTTM)
September 16, 2010

Justin E. Gottschlich, Jeremy G. Siek, Manish Vachharajani
University of Colorado-Boulder
InvalSTM

- Scan active transactions for conflicts
- Version-less memory
- Visible readers (bloom filter-based)
- Deferred update
- Lazy conflict detection / resolution
InvalSTM Goals

• Performance
  o CGO’10

• Correctness
  o WTTM’10
  o … ?
Theoretical Approach

• Define
  o InvalSTM automaton (Lynch et. al’s)

• Extend
  o Conflict graph, conflicts, and more (Papadimitriou)
• \(<T \text{ abort}\> \quad o \quad T \text{ aborts}

• \(<T, x.\text{read}(v)\> \quad o \quad T \text{ reads variable } x

• \(<T \text{ commit, } I, W\> \quad o \quad T \text{ commits}
  o \quad I \text{ is a set of invalidated transactions}
  o \quad W \text{ is a set of ordered pairs } (x, v)
Lazy Conflict Graph

• Similar to Conflict Graph
  o transactions are vertices
  o ordering for conflicts are directed arcs
  o conflict serializable iff acyclic

• Different from Conflict Graph
  o assumes deferred update
  o no write events, just commits
  o \(WW\) and \(WR\) conflicts are always serializable
Conclusion

• Proof for InvalSTM
• Theoretical framework for deferred TM systems
• Next up:
  o lock-aware TM semantics
Questions?

Justin E. Gottschlich

justin.e.gottschlich@raytheon.com
http://eces.colorado.edu/~gottschl/

Raytheon
Colorado
University of Colorado at Boulder