CSE548 – Lecture 12
Transactional Memory

Thierry Moreau
TM: Architectural Support

• Transactional memory: efficient support for lock-free synchronization
  – Avoids: Priority inversion, convoysing, deadlocks
• Two properties of TM:
  – Serializability: appear to execute serially
  – Atomicity: transactions either commit or abort
• Primitives for accessing memory:
  – Load-transactional, Load-transactional-exclusive, store-transactional
• Primitives for manipulating transaction state:
  – Commit, abort, validated
  – Commits succeed only if no other transactions has updated the memory location’s data set, and no other transaction has read any location in the transaction’s write set.
TM: Architectural Support

• Implementation:
  – Hardware support is restricted to primary caches
  – TM is implemented by modifying standard multiprocessor cache coherence protocols
  – At any time, a memory location is either:
    • Not immediately accessible by any processors
    • Accessible non-exclusively by one or more processors
    • Accessible exclusively by exactly one processor
  – Two caches: a regular cache for non-TM ops, and a transactional cache for transactional ops
  – Transactional cache holds all the tentative writes, w/o propagating them to the other processors or to main memory unless the transaction commits
Virtualizing TM

• Issues with architectural support for TM:
  – Hardware proposals requires programmers to be aware of platform-specific resource limitations
  – Examples are: buffer sizes, scheduling quanta, page faults, process migration

• VTM:
  – Solves resource or scheduling limitation issues
  – Use programmer transparent software structures to hide the buffer overflow, page faults, context switches etc.
  – The idea is to decouple transactional state from processor state
Discussion

• Open challenges in TM
  – Virtual address aliasing to allow sharing between two processes
  – OS sys calls within a transaction
  – I/O within a transaction?