Toward Transactional Memory Workloads

Ali-Reza Adl-Tabatabai

Microprocessor Technology Labs
What primitives can workloads assume?

Language-level construct to delimit atomic blocks

Too tedious to manually instrument memory accesses
→ let the compiler insert & optimize the STM calls

- Managed language (e.g., Java): atomic {...}
- C/C++: #pragma txn {...}, txnBegin()...txnCommit()

Condition synchronization primitive

- retry, when, watch, or/orelse
What else to assume?

Semantics & restrictions
- Allow I/O
- Provide strong atomicity
- Interoperate with locks, threads & precompiled libraries

Tools
- Debuggers
- Performance analysis & diagnostics
To what extent will available systems bias the applications?

Applications developed on SMPs will not scale to higher levels of HW parallelism

→ That’s bad for evaluating scalability of TM implementations

Need a mixture of workloads
  - Workloads with contention to stress contention management
  - Scalable workloads to stress TM implementation scalability
XviD scalability on a 16-core CMP

Similar issues will arise in TM implementations
Need workloads that scale to stress TM implementation

Differences manifest only beyond 4 cores
XviD scalability on a 32-core CMP

Going from 16 to 32 cores changes the curves
MSPC 2006
Call for papers

Memory Systems Performance & Correctness

Co-located with ASPLOS, Oct 22, 2006, San Jose, CA

Paper submission deadline: July 28, 2006

General chair: Antony Hosking

Program Committee:
Emery Berger           Amer Dian
Hans Boehm            Cormac Flanagan
Martin Burtscher      Thomas Gross
Barbara Chapman    Dan Grossman
David Chase            Glenn Reinman
Brad Chen              Qing Yi