CS184b: Computer Architecture (Abstractions and Optimizations)

Day 25: May 27, 2005
Transactional Computing

Today
• Transactional Model
• Hardware Support
• Course Wrapup

Transactional Compute Model
• Computation is a collection of atomic transactions
• Transactions are performed in some ordering
  – Specified sequential ordering
  – Some consistent sequential interleaving
• Looks like sequential (multithreaded) computation

Atomic Blocks
• Indivisible
• Cannot get interleaving of operations among blocks
• Each block
  – reads state at some point in time;
  – provides all its updates (writes) at that point

How Get?
• How do we get atomic blocks in our current models?
• Get parallelism with this model?

Simple/Sequential Version
• Acquire a mutex
• Perform transaction
• Release mutex
Parallel/Locking Version

- Read inputs, locking each as encounter
- Lock all outputs
- Perform operations
- Write outputs
- Unlock inputs/outputs

- OneChip Vector case is a single source/sink version of this

Parallel Locking Dangers

What do we have to worry about in this version?

- Lock acquisition order → deadlock
- Parallelism?

Optimistic (Speculative) Transactions

- **Pattern:** speculate/rollback
  - Common case: non-interacting
- Read inputs
- Perform computations
- Check no one changed inputs
  - Got correct inputs? → write outputs atomically
  - Inputs changed → rollback and try again

Optimistic Transactions

- Compare
  - advanced load
  - ll/sc
  - branch prediction
- Doesn’t need to be sequentialized
- Just need to look sequential
- It is adequate to
  - detect sequentialization violation

Hardware Microarchitecture

- What need?
  - Keep/mark inputs
  - Hold on to outputs
  - Check inputs
  - Commit outputs
Microarchitecture

- Mark inputs in processor-local cache
- Writes into processor-local cache
- Write buffer for outputs
  - Hold until end of transaction
- Commit bus
- Snoop on bus for invalidations
- Dump write buffer to commit bus when turn to commit

uArch Concerns?

- Impact of commit sequentialization?
- Time spent rolling back?
- Size of cache/write-buffer needed?

Commit Traffic

Where time go?

Read State
Write State

![Graph showing write state](image)

[Hammond et al./ISCA2004]

Execution Time

![Graph showing execution time](image)

[Hammond et al./ASPLOS2004]

Other Patterns?

- Double Buffer
- Parallel Prefix for Commits

Programming

- Explicit
  - Parallel/transactional looping constructs
  - forks
- Add like futures
  - Force ordering to match sequential language spec and is "safe"
- Automatically add
  - Support "Mostly Functional" programming
  - "Make the fast case common"

Performance Issues

- Too many violations?
  - Force some transactions to wait on others
    - Explicit/pragma/program construct
    - Feedback-based
      - Empirical dataflow discovery?

Stanford TCC Prototyping

(from Workshop on Architecture Research using FPGA Platforms at HPCA 2004)
ATLAS Overview

- A multi-board emulator for transactional parallel systems

Goals
- 16 to 64 CPUs (8 to 32 boards)
- 50 to 100MHz
- Stand-alone full-feature system
  - OS, IDE disks, 100Mb Ethernet, ...

ATLAS architecture space
- Small, medium, and large-scale CMPs and SMPs
- UMA and NUMA
- Flexible transactional memory hierarchy & protocol
- Flexible network model
- Flexible clocking, latency, and bandwidth settings

Example: 2-way bus-based transactional CMP

![Diagram of 2-way bus-based transactional CMP]

Big Ideas [Transactional]
- Model/Implementation
  - Only needs to appear sequentialized
- Speculation/Rollback
- Common Case

MODEL REVIEW

Models
- Single
  - Conv. Processors
- Multiple data
  - Parallel (SIMD, Vector)
- Pure MP
- Trans- Actional Dataflow (e.g. SCORE)
- Fine-grained threading
- Shared Memory
- Side Effects
- No Side Effects
- No SM
- DSM
- SM MP
- (S)MT

Building Block: Xilinx ML310 Board

- XC2V30 FPGA features
  - 2 PowerPC 405 cores
  - 2.4Mb dual-ported SRAM
  - 30k logic cells
  - 8 RocketIO 1.25Gbps transceivers

- System features
  - 256MB DDR, 512MB CompactFlash
  - Ethernet, PCI, USB, IDE, ...

- Design and development tools
  - Foundation ISE for design entry, synthesis, ...
    - For the transactional memory hierarchy and network
  - Chipscope Pro logic analyzer for debugging
  - EDK for system simulation, system SW development, configuration, ...
  - Montavista Linux 3.1 Pro

Caltech CS184 Spring 2005 -- DeHon
Big Ideas

• Model
• Implementation support model semantics
  – But may be very different, optimized
• Scaling and change
• Common case fast; fast case common
  – Measure to understand