A Comparison of Two Approaches to Parallel Simulation of Multiprocessors

Andrew Over*, Bill Clarke, and Peter Strazdins

CC-NUMA Project,
Department of Computer Science,
The Australian National University

ISPASS 2007, 26 April 2007

1 Overview

- Motivation
- Simulator background and design
- Parallelization approaches:
  - via locking
  - via PDES techniques
- Performance comparison:
  - Methodology and workloads
  - Simulator metric stability
  - Performance comparison
  - Load balancing concerns
- Conclusions
2 Motivation

- CMP systems are now widely available:
  - Multi-core simulation is now needed in more situations
  - Common platforms are suitable hosts for parallel simulation
- Serial simulation of MP or CMP systems incurs a linear slowdown
- Obtaining results quickly is sometimes more useful than obtaining results efficiently
- Difficult to parallelize owing to degree of communication between processors on shared-memory platforms
3 Simulator Infrastructure

- Models multiprocessor UltraSPARC IIICu systems
  - In-order superscalar processor
  - 32KB I-cache, 64KB 4-way pseudo-random D-cache
  - 1–8MB 2-way E-cache
- Execution driven (fetch-decode-execute loop)
- Memory system model (including FirePlane interconnect protocol)
- Runs unmodified Solaris binaries using a syscall emulation layer
- Typical slowdown 500–1000 ×
- Simulation target Sun V1280:
  - 12 × 900MHz UltraSPARC IIICu
  - 24GB RAM, single snooping coherence domain
- Simulation target doubles as Simulation host
4 Simulator Implementation Details

- Memory hierarchy is the key obstacle to parallelization
  - All processor interactions occur through cache coherence
  - Coherency protocol must operate in parallel but preserve correctness
- Processor model isolated from memory hierarchy by a *bridge*, which interacts with the *backplane* model
- Two different interconnect models:
  - Simple approximate model (*passive*)
  - Detailed complex model (*active*)
5 Simulator Passive Backplane

- Passive backplane is a simple interconnect model:
  - Fixed request latency based on type and coherency result
  - Coherency managed by directly inspecting and modifying foreign caches
  - No queueing of cache events (handled immediately)
- Serial multiprocessor simulation employs round-robin timeslicing
- Multiple simultaneous accesses $\Rightarrow$ inconsistent state
- Host locking used to protect data structures
- Anecdotally 20% of instructions reference memory $\Rightarrow$ a single lock would cripple performance
6 Simulator Passive Backplane (cont)

- Lock chosen according to cache index to prevent conflict
- No lock required on D-cache load hit
- Lock required on D-cache store, E-cache access
- Lock selection restricted to PA bits in D, E-cache indexes (128 locks)
- Majority of memory references remain unlocked
- Host locking introduces non-determinism
- Strict temporal ordering of requests not guaranteed
7 Simulator Active Backplane

- Detailed event-driven model of the interconnect:
  - All coherency states modelled (including transient states) based on FirePlane protocol
  - Cache coherency implemented via snoop request and snoop response events
- A minimum latency exists between event initiation and visibility on foreign processors (7 bus cycles @ 150MHz)
- Conservative parallel discrete event simulation techniques (PDES) may be applied
- Processors may run for limited periods without communication, before exchanging events
8 Simulator Active Backplane (cont)

- Parallel simulation consists of alternating serial and parallel phases:
  - Backplane runs in isolation
  - Processor models run in parallel interacting only with backplane

- Backplane arbitrates and queues events for processors
- Processor model handles queued events, forward responses
- All communication handled during serial phase
- Parallel processor simulation with complete determinism
- User-space barriers employed to synchronize at the end of a timeslice
9 Methodology

- Parallel simulation of workloads normalized against serial simulation of an identical workload
- Two synthetic benchmarks used:
  - `spintest` – Tight loop, no memory references
  - `sharetest` – Frequent stores to small shared region
- Selected small scale NAS Parallel Benchmarks used
- Evaluating simulator performance, not simulation target performance
- Many simulation configurations tested (differing target processors per thread)
10 \textit{spintest}

![Graph showing Simulator Speedup (spintest)]
11 sharetest

Simulator Speedup (sharetest)

- active backplane
- passive backplane

Simulated CPUs

Speedup (vs serial)
12  \textit{is.} W

![Graph showing Simulator Speedup (is.W) for active and passive backplanes. The x-axis represents the number of simulated CPUs, ranging from 1 to 32. The y-axis shows speedup versus serial, ranging from 1 to 6. The graph includes data points for both active and passive backplanes, illustrating the speedup improvement as the number of simulated CPUs increases.]
13 Passive Backplane Stability

![Graph showing Passive Backplane Stability]
Load Balancing

- Active backplane has no locking or contention
- In spite of barrier overhead, serial component, speedups well short of ideal
- Speedup strongly depends on workload
- Analysis shows constraint is per-thread load balancing:
  - Simulation time for one interval depends largely on CPI
  - Variation in CPI across threads introduces load imbalance
- Multiple simulated processors per thread reduces the severity
- A more complex processor model (employing fewer abstractions) may be less susceptible to this problem
15 Passive Backplane Stability

![Graph showing normalized frequency vs. single processor timeslice simulation (ns).]

- 1 simulated cpu/host thread (red)
- 2 simulated cpu/host thread (blue)
- 4 simulated cpu/host thread (magenta)

The graph illustrates the stability of passive backplane simulation for different numbers of simulated CPU/host threads.
16 Conclusions

- (Small scale) parallelization of architectural simulation is possible
- Non-deterministic parallelism conceptually simple with slight dependence on target architecture
- Deterministic parallelism is more involved and exploits interconnect properties
- Non-determinism introduces concerns over result stability
- Observed speedups depend on workloads and constrained by load-balancing of simulated processors
Future Work and Acknowledgements

- **Future Work:**
  - Additional analysis of stability concerns (including serial round-robin interleaving)
  - Evaluation of larger workloads, different architectures

- **Acknowledgements:**
  - CC-NUMA Project
  - Australian Research Council
  - Sun Microsystems
  - Gaussian Inc
  - ISPASS