SAMPLED SIMULATION OF MULTI-THREADED APPLICATIONS

TREVOR E. CARLSON, WIM HEIRMAN, LIEVEN ECKHOUT

HTTP://WWW.SNIPERSIM.ORG
MONDAY, APRIL 22ND, 2013
ISPASS 2013, AUSTIN, TX
Flex Points / COTSon’s Dyn. Sampling
OVERVIEW

• How can we help the hero save the princess?
• How can we create a representative sample of a multi-threaded application?

• Prior Work
• Key Contributions of this Work
• Results and Evaluation
DEMANDS ON SIMULATION ARE INCREASING

- Increasing cache sizes
  - Simulation requires realistic application working sets
  - Scaled-down applications might not exhibit the same behavior

- Increasing core counts

- Multi-threaded workloads

- New solutions are needed
**WORKLOAD REDUCTION IS THE KEY**

- Many workload reduction techniques exist today
  - Reduction
    - Smaller input sizes
    - Reduced numbers of iterations
  - Sampling: only part of the workload needs to be simulated in detail, whole-program performance can be extrapolated
    - SimPoint
    - SMARTS
    - FlexPoints
Sampling Multi-Threaded Workloads

• Define: synchronizing multi-threaded application
  – Use locks (mutexes), barriers, etc.
  – Application where multiple threads are working to solve a problem together

• Multi-threaded application complexities
  – We want to determine application runtime, not CPI
  – Can be different performance per thread (e.g. NUMA, load imbalance)
  – Instruction count cannot be used to determine fast-forward length (per-thread CPI, thread idle time)
MULTI-THREADED SAMPLING

• Goal
  – Reduce multi-threaded application simulation time
  – Accurately predict application runtime

• Key Contributions
  – Sampling in time is a requirement for sampling simulation of multi-threaded applications
  – Take into account thread details during fast-forwarding
    • Thread synchronization (mutexes, barriers, etc.)
    • Per-thread CPI
  – Application phase behavior is critical for accurate sampling
CURRENT SAMPLING SOLUTIONS

• Current multi-threaded solutions are not sufficient
  – Flex Points
    • Specifically designed for non-synchronizing throughput (server) workloads
    • Issue: Assumes no correlation between threads
  – COTSon’s Dynamic Sampling (Argollo et al., Ryckbosch et al.)
    • Issue: Doesn’t properly handle synchronization during fast-forwarding

Wenisch, et al., IEEE MICRO 2006
Argollo et al., ACM SIGOPS Operating Systems Review
MULTITHREADED FAST-FORWARDING

• Use time as the base unit for sampling
  – Time is common across threads, unlike instructions

• Use instruction count as a low-overhead fast-forwarding method
  – Functional-execution only provides instruction count, but we still require time for fast-forwarding

• Use per-thread non-idle CPI from previous detailed interval
MULTITHREADED FAST-FORWARDING

• Propagate time from waker to waiter (as in detailed)
• Only need instruction count during fast-forwarding
  – Efficient implementation in Pin with multiple instr. modes
  – Maintain time using instruction count and per-thread IPC
It is possible to get good accuracy at high speed, but not reliably.
APPLICATIONS ARE PERIODIC

nbp-ft, class A, 8 threads
**Main Problem: Aliasing**

- When application exhibits periodicity near detailed interval length, aliasing errors

```
<table>
<thead>
<tr>
<th>IPC</th>
<th>Ne</th>
<th>O</th>
<th>P</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- New problem to multi-threaded sampling:
  - SMARTS uses >10,000 sampling units: average IPC is obtained
  - SimPoint sampling units can still alias application periods
  - Key insight: we need single sample accuracy for fast-forward IPC

- Sampling parameters determined by application periodicity
IDENTIFY PERIODICITIES

• Application periodicities are identified in a micro-architectural independent manner

BBV Autocorrelation
npb-ft, class A, 8 threads, with 550k and 1.14M insn periodicities

OMP Call Structure
npb-lu, class A, 8 threads with high variability (not used)
IDENTIFY PERIODICITIES

• We do this in an architecture-independent way
• Sampling sufficiently above or below the period will minimize error

D = Detailed period
F = Fast-forward (multiple of D)
**Sampling Process**

- Sampling sufficiently above or below the period will minimize error
**Experimental Setup**

- **Sniper Multi-core Simulator**
  - Nehalem-style architecture
    - 2 sockets, 4 cores per socket
    - 2.66 GHz, 128-entry ROB
    - 32 KB L1-I, 32KB L1-D, 256 KB L2/core, 8MB L3/4 cores

- **Benchmarks**
  - NAS Parallel Benchmarks 3.3.1, class A inputs
  - Parsec 2.1, simlarge input set
  - SPEC OMP2001, train input set
Even with oracle per-thread CPI knowledge up-front, our proposed methodology provides a more accurate solution.
RESULTS

• Predicted Most-Accurate Results
  – Average absolute error of 3.5%
  – Average speedup of 2.9x, maximum of 5.8x
Multi-threaded Sampling

• Key Contributions
  – Sampling in time is a requirement for sampling simulation of multi-threaded applications
  – Take into account thread details during fast-forwarding
    • Thread synchronization
    • Per-thread CPI
  – Taking into account application phase behavior is critical for accurate sampling

• Predicted Most-Accurate Results
  – Average absolute error of 3.5% across applications
  – Average speedup of 2.9x, maximum of 5.8x
**Multi-Threaded Sampling Release**

• Sniper 5.0 Release

– Multi-threaded sampling infrastructure

– Available from:
  • [http://snipersim.org](http://snipersim.org)

Interval core model, CPI-stacks, advanced visualization support, automatic topology generation, parallel multi-threaded simulator, multi-program and multi-threaded application support, x86 and x86-64 support, hardware validated, full DVFS support, shared and private cache support, scheduling support, heterogeneous configuration, modern branch predictor, OpenMP, MPI, TBB, OpenCL, integrated benchmarks, SPLASH-2, most of Parsec, McPAT integration, SimAPI, Python scripting, single-option debugging, modern OS support, Pin-based, statistics database, stackable configurations
SAMPLED SIMULATION OF MULTI-THREADED APPLICATIONS

TREVOR E. CARLSON, WIM HEIRMAN, LIEVEN EECKHOOUT

HTTP://WWW.SNIPERSIM.ORG
MONDAY, APRIL 22ND, 2013
ISPASS 2013, AUSTIN, TX