

Evaluating Non-deterministic Multi-threaded Commercial Workloads

**Alaa R. Alameldeen, Carl J. Mauer, Min Xu,
Pacia J. Harper, Milo M.K. Martin, Daniel J. Sorin,
Mark D. Hill, and David A. Wood**

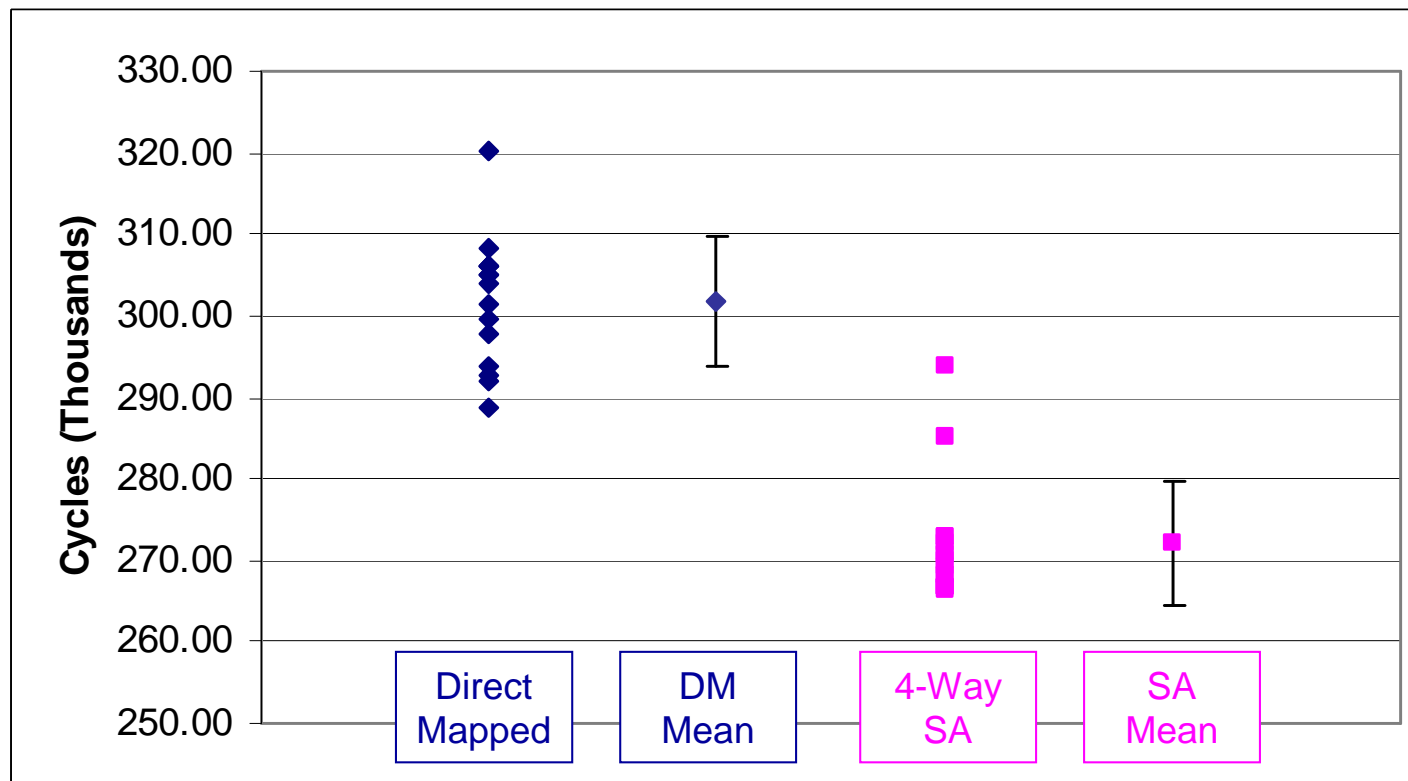
Computer Sciences Department
University of Wisconsin—Madison
<http://www.cs.wisc.edu/multifacet>

Introduction

- Short measurements on real machines require multiple runs
 - Uncontrolled factors
 - Want to separate random from systematic effects
- Simulation measurements use a single run
 - Simulators are deterministic
 - No uncontrolled factors
- **Wrong!**
 - Multi-threaded workloads can be **unstable**
 - Small changes in timing cause large changes in results

Introduction

- Instability may affect conclusions
 - Comparing Direct Mapped to Set-Associative Caches



Evaluating Non-deterministic Multi-threaded Commercial Workloads

Overview

- Introduction
- Methods
- Workloads
- Result I: Process scheduling
- Result II: Workload Variability
- Conclusion
- Future Work

Methods

- Real machine
 - Setup, tune, validate on a 16-processor Sun E6000
 - 8 – 16 X speed-up for each application
- Simulator
 - Simics, Full-system simulator running Solaris 8
 - Ruby, Memory timing simulator
- Experiments
 - Start from a warm checkpoint
 - Measure throughput (transactions completed / time)

Workloads

- OLTP
 - TPC-C-like benchmark using a 1 GB database
- SPECjbb
 - Server-side Java-based middleware workload
- Apache
 - Static web serving: Apache driven by SURGE
- Slashcode
 - Dynamic web serving message board, using code and data similar to `slashdot.org`

Why unstable?

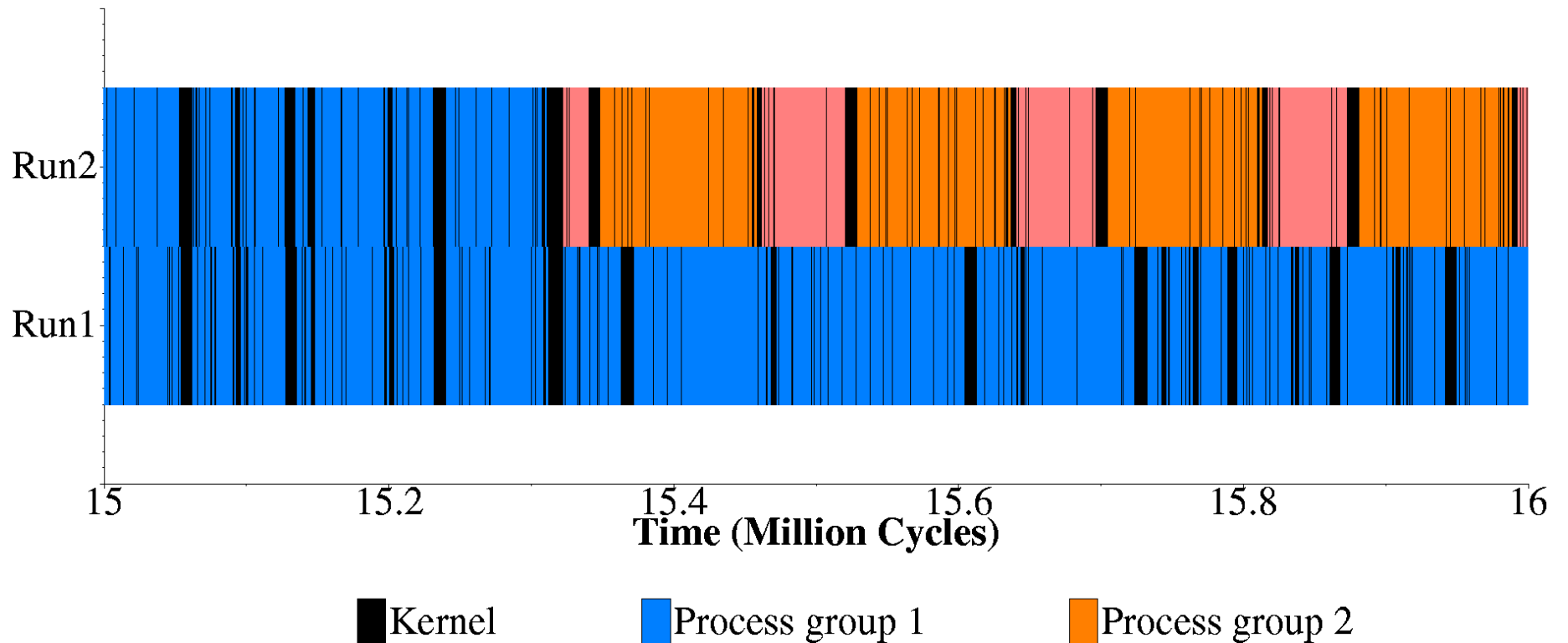
- Different paths are executed
- Hypotheses
 - Process scheduling
 - Order of lock acquisition

Result I: Process scheduling

- Deterministic simulation of OLTP on *uniprocessor*
- Artificially injected misses to I-cache
 - Run1: 0, 100, 200 ...
 - Run2: 50, 150, 250 ...
- Measured equivalent to 3-5 seconds in real system
- Run time difference of 9%
- Is process scheduling a factor?

Result I: Process scheduling

- Traced process groups scheduled on CPU

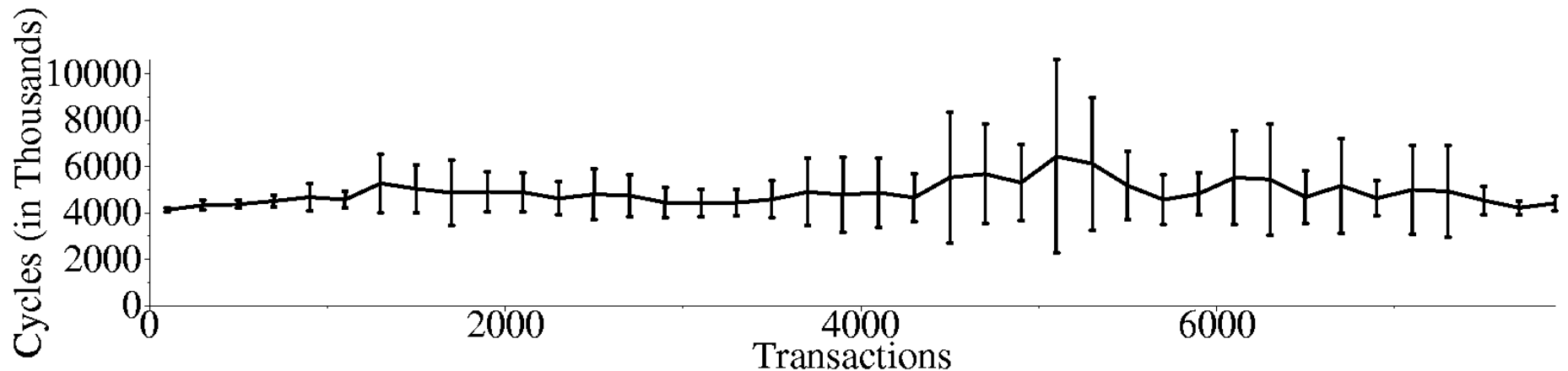


Methods, part II

- Pseudo-random perturbations
 - Run multiple runs from same checkpoint
 - All runs have same average memory latency
 - Misses to main memory perturbed by 0-4%
- Calculate mean, standard deviation

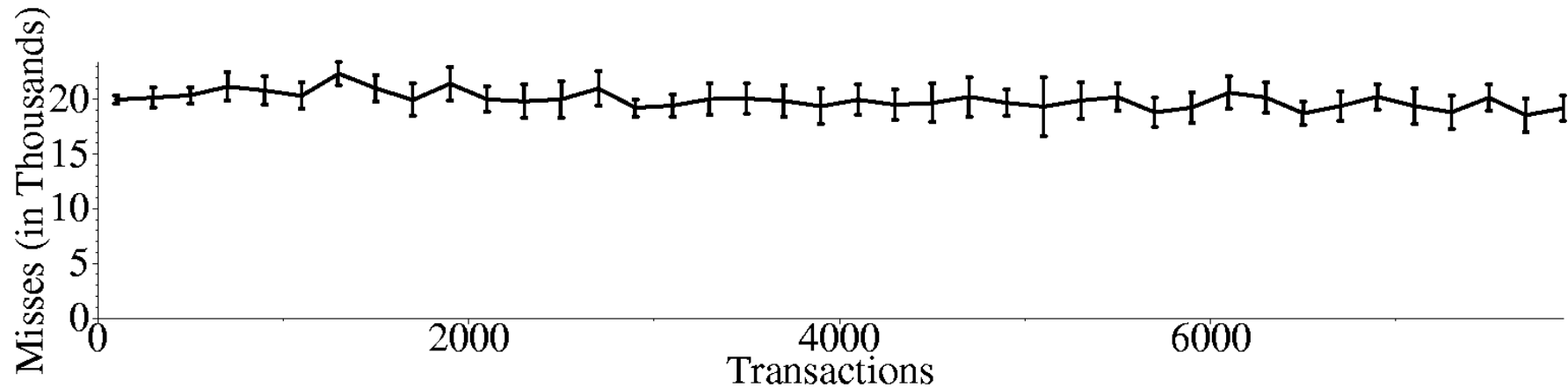
Result II: Variability

- Variability
 - 16-processor system running 8,000 OLTP transactions
 - 20 runs from same checkpoint
 - 12 – 20 seconds in real system
- $1 / \text{Throughput}$ (cycles per transaction)



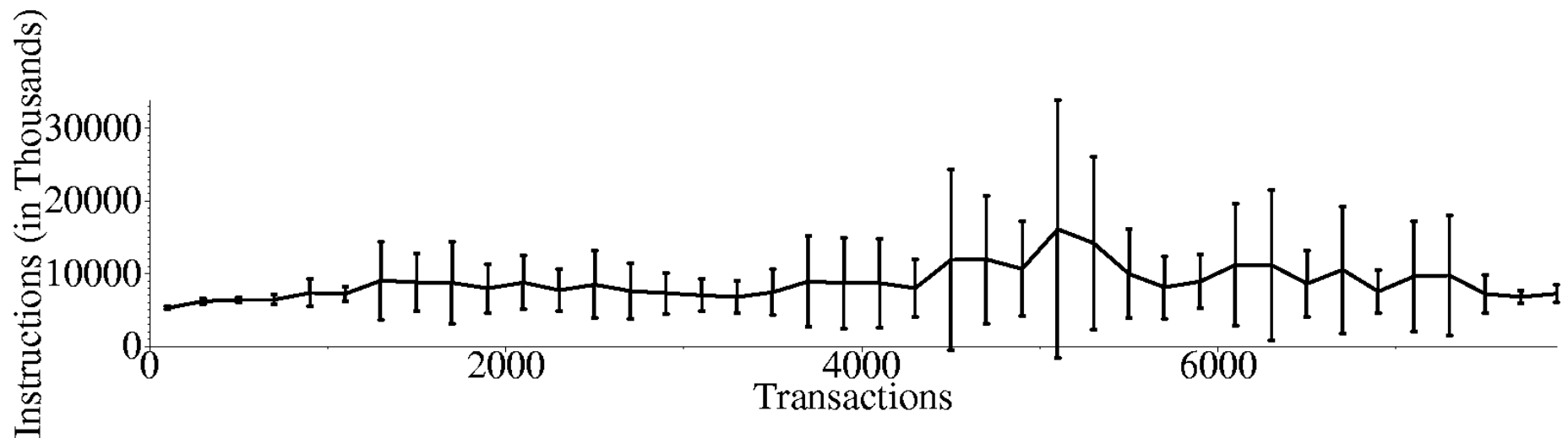
Result II: Variability

- Miss rate (misses per transaction)

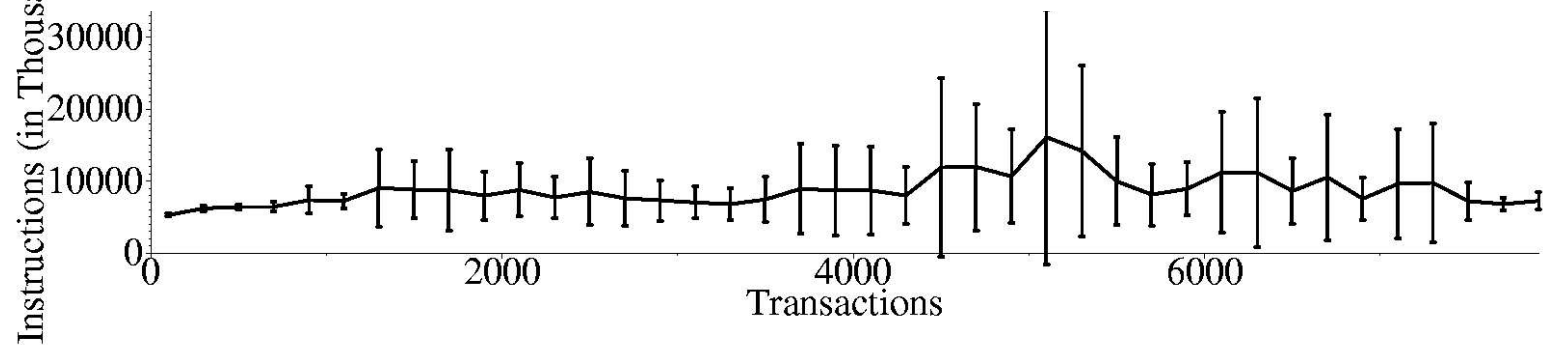
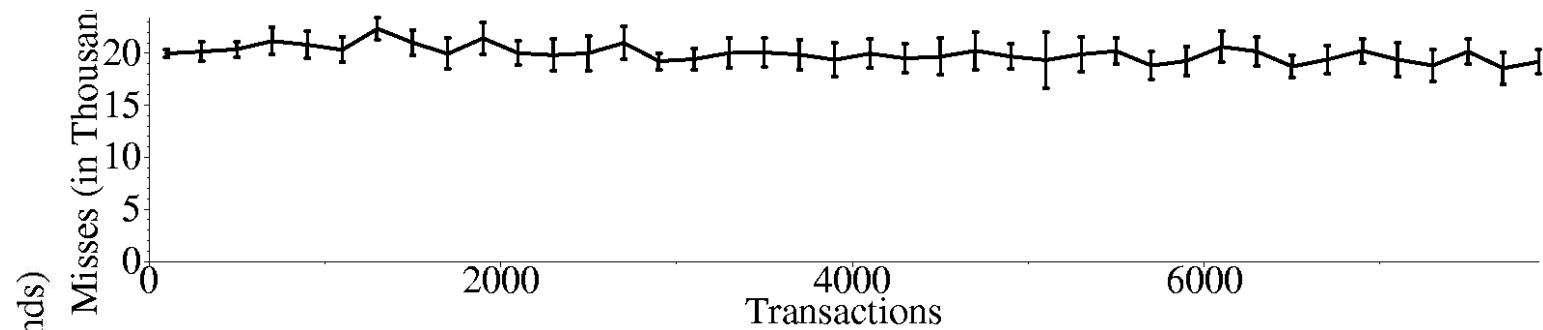
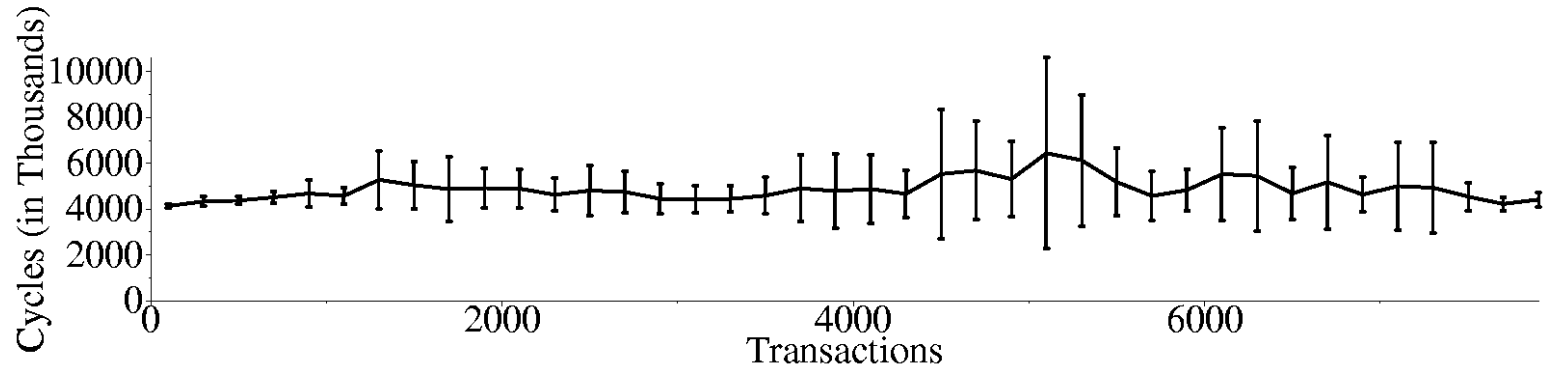


Result II: Variability

- Instructions executed (per transaction)
- Hypothesis
 - “Spin-waiting” hypothesis
 - Lock-acquisition, idle loop, device activity



Result II: Variability



Conclusion

- Multi-threaded commercial workloads can be unstable even on uniprocessors
- Instability can affect conclusions in short runs
- Pseudo-random methodology can help
- Even within one workload variations exist

Future Work

- Root cause(s)?
- Methodology improvements
- Quantify instability further

Questions

Evaluating Non-deterministic Multi-threaded Commercial Workloads