# Variability in Architectural Simulations of Multi-threaded Workloads

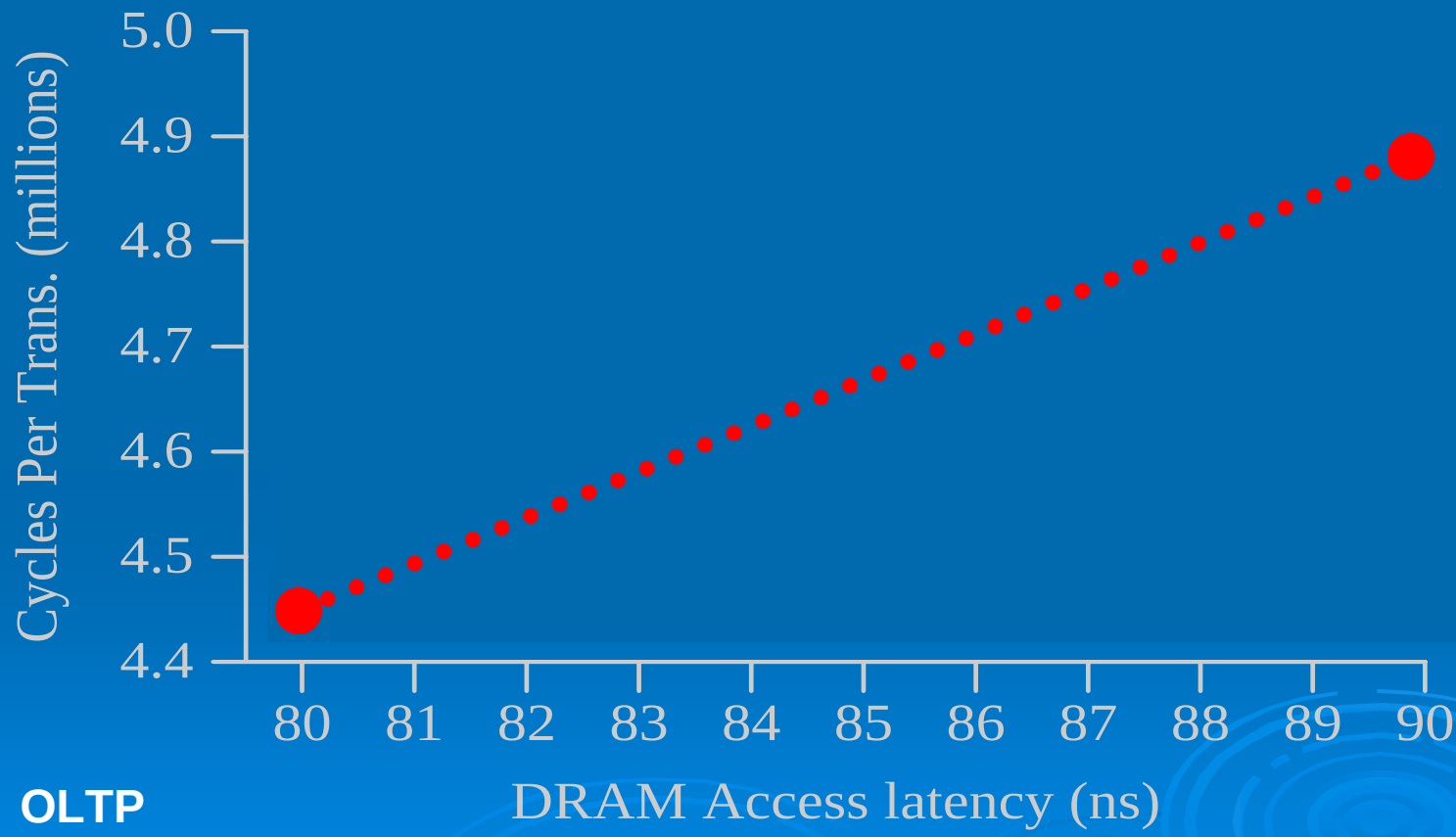
Alaa R. Alameldeen and David A. Wood
University of Wisconsin-Madison
{alaa,david}@cs.wisc.edu
http://www.cs.wisc.edu/multifacet/

# Motivation

- Experimental scientists use statistics
- Computer architects in simulation experiments don't!
- Why ignore statistics?
  - Simulations are deterministic

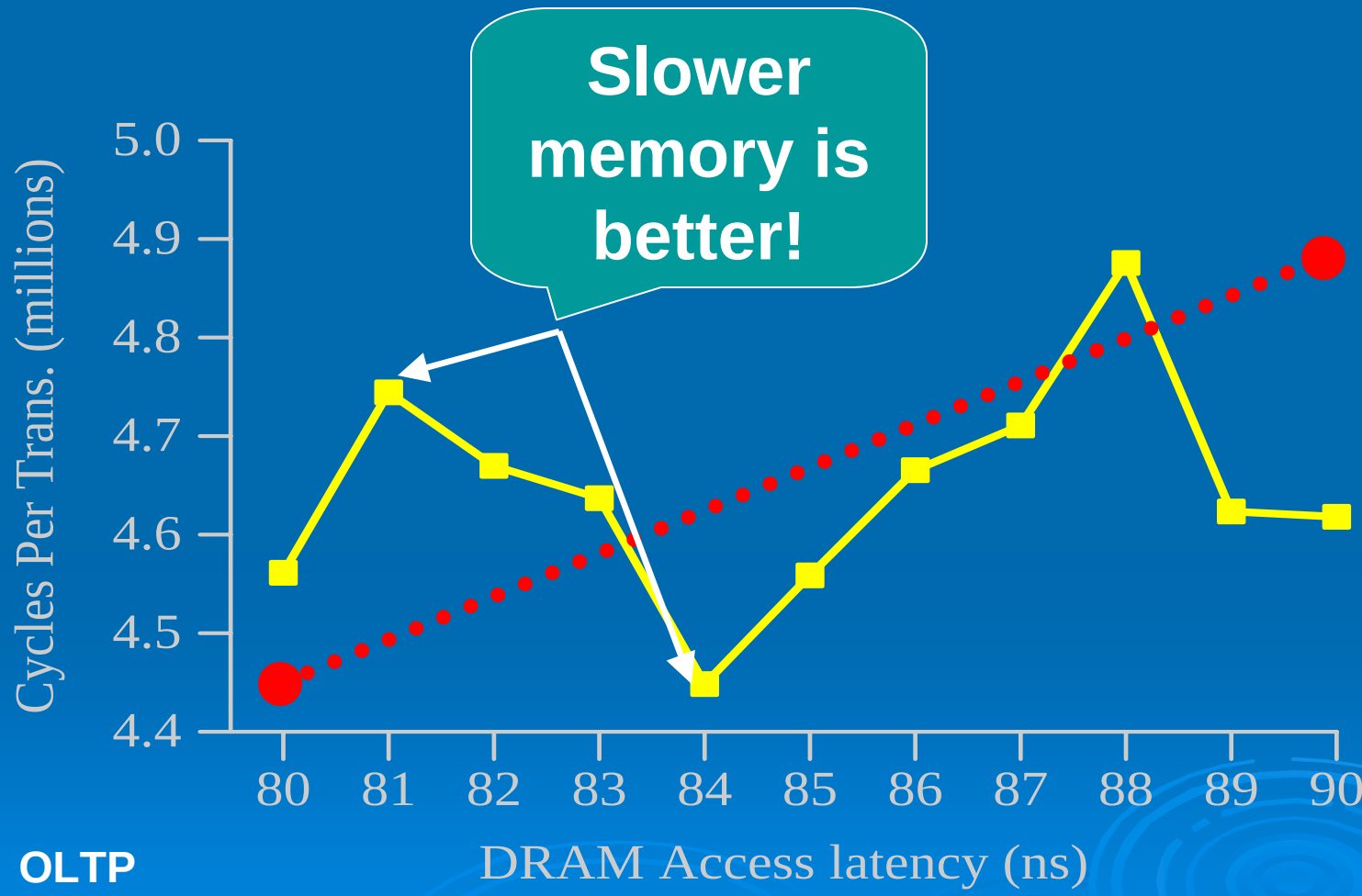
➲ This can lead to wrong conclusions!

# Workload Variability




**OLTP**

# Workload Variability




**OLTP**

# What Went Wrong?

- Many possible executions for each configuration
- Why? Different timing effects
  - OS scheduling decisions
  - Different orders of lock acquisition
  - Different transaction mixes
- This is magnified by short simulations

➲ Variability can lead to wrong conclusions

# Overview

- Variability is a real phenomenon for multi-threaded workloads
  - Runs from same initial state can be different
- Variability is a challenge for simulations
  - Simulations are short
- Our solution accounts for variability
  - Multiple runs, statistical techniques

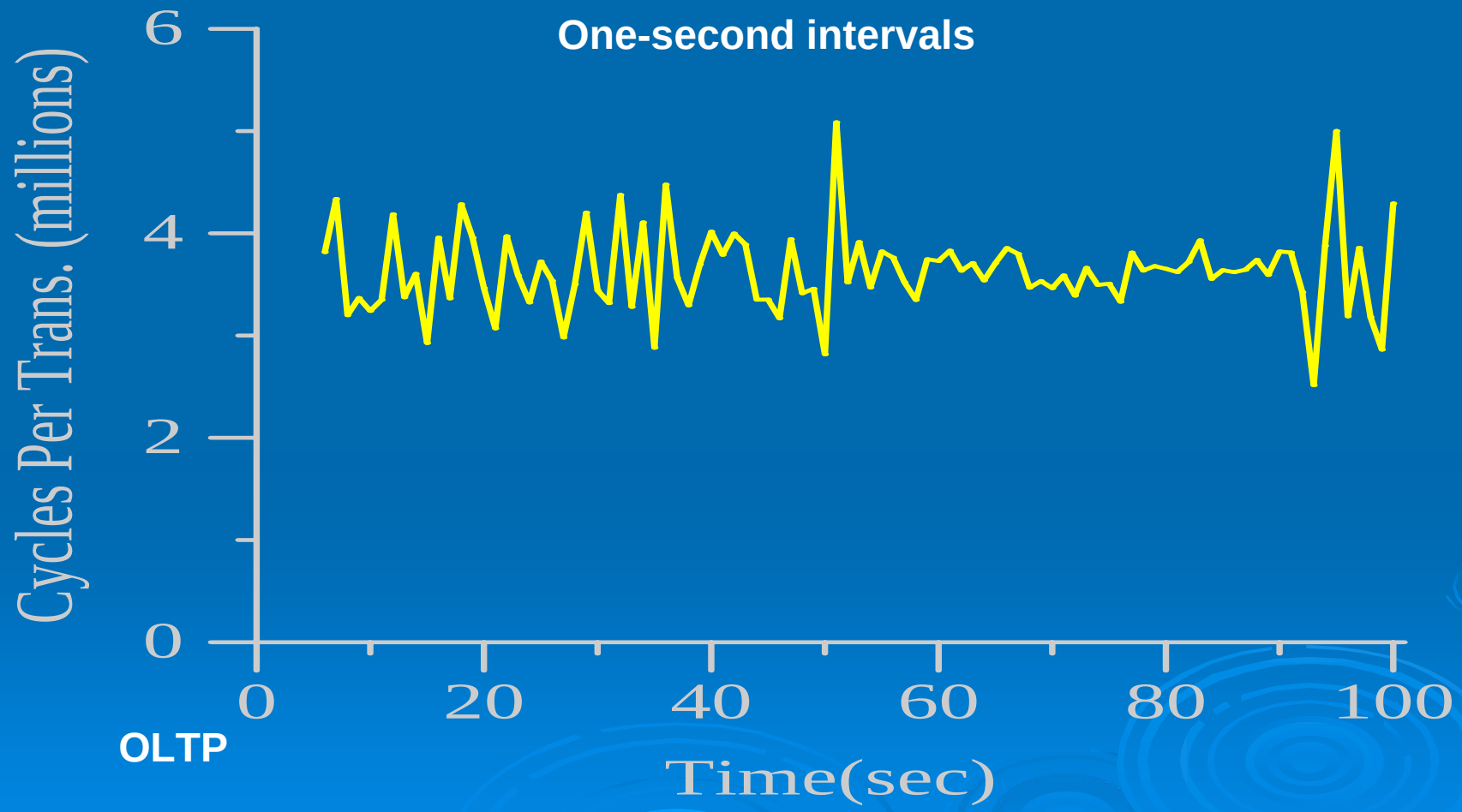
# Outline

- Motivation and Overview
- Variability in Real Systems
  - Time and Space Variability
- Variability in Simulations
- Accounting for Variability
- Conclusions

# What is Variability?

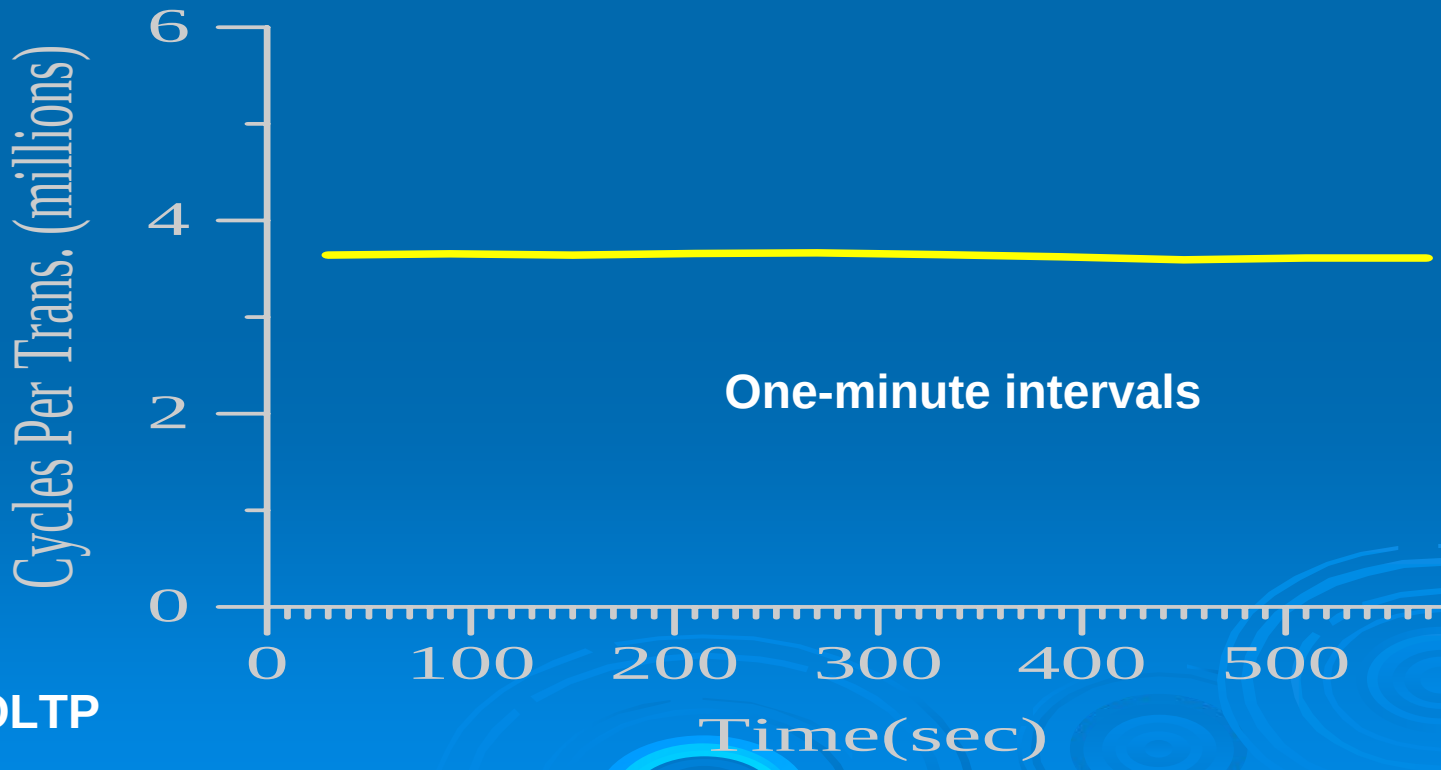
- Differences between multiple estimates of a workload's performance
- Time Variability:
  - Performance changes during different phases of a single run
- Space Variability:
  - Runs starting from the same state follow different execution paths

# Time Variability in Real Systems

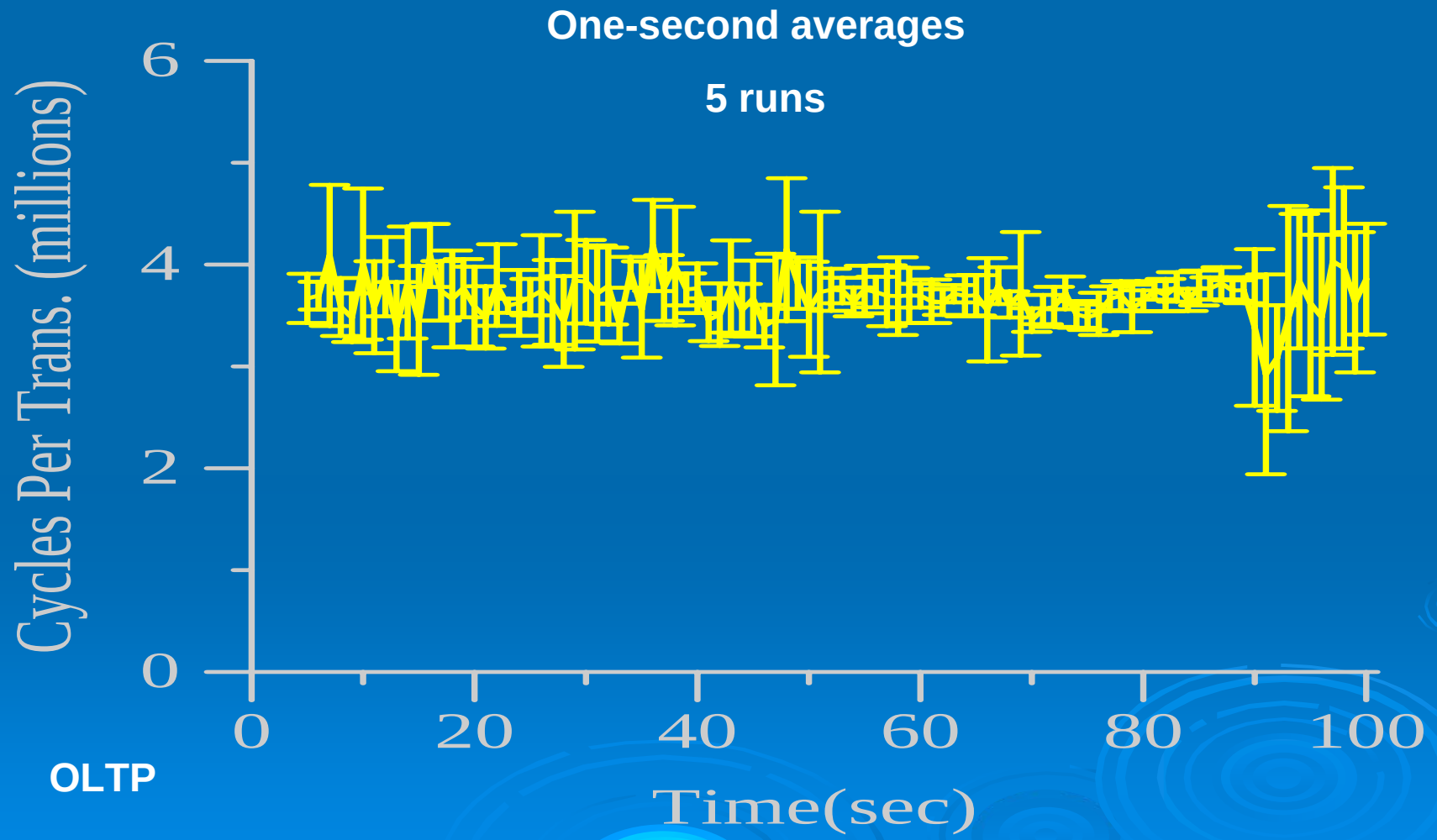


# Time Variability Example (Cont'd)

- How is this handled in real experiments?
  - Solution: Run your experiment long enough!



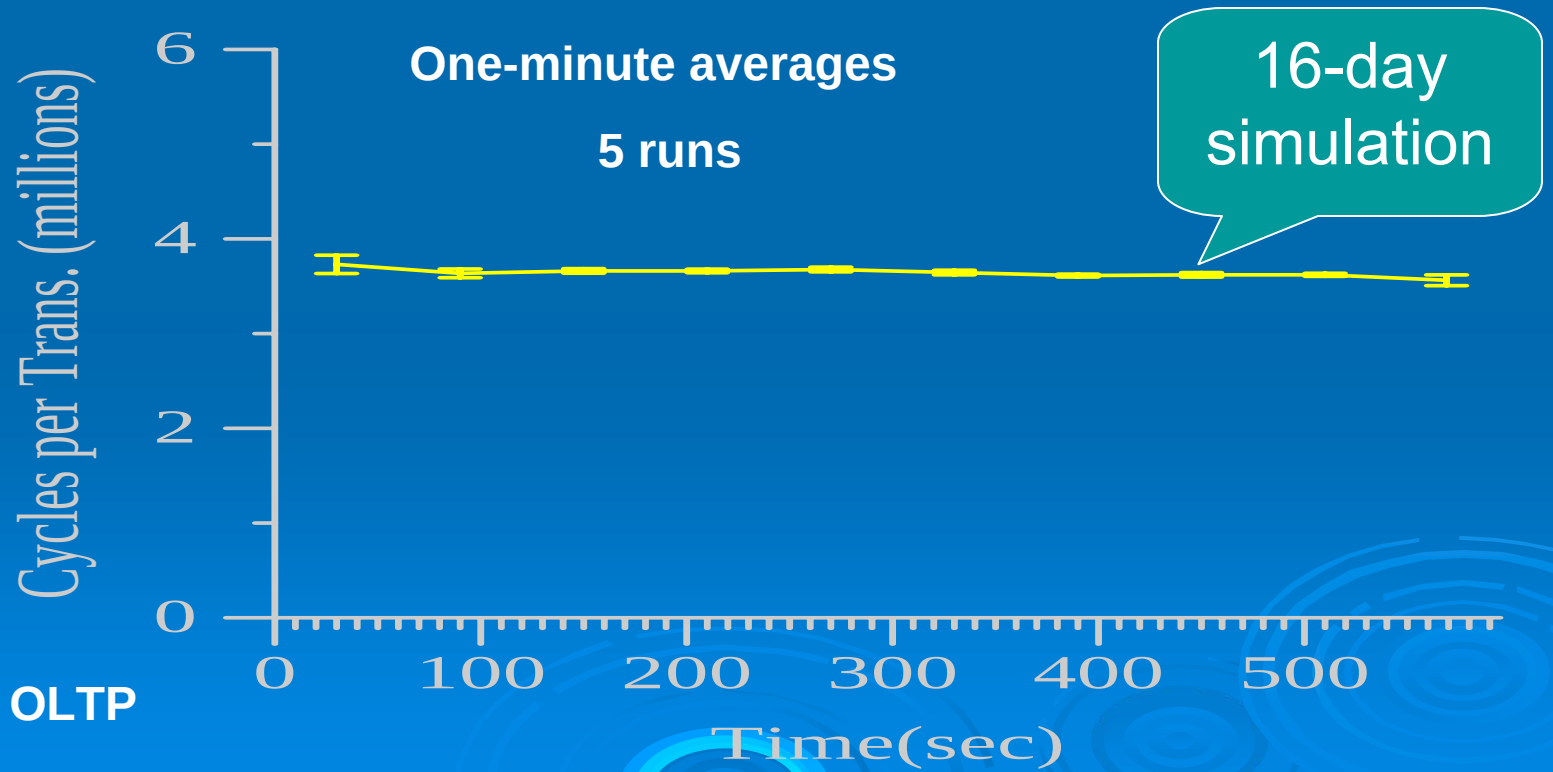
# Space Variability in Real Systems




**OLTP**

# Space Variability Example (Cont'd)

- How is this handled in real experiments?
  - Same Solution: Run your experiment long enough!



# Outline

- Motivation and Overview
- Variability in Real Systems
- Variability in Simulations
  - Simulation Infrastructure
  - Injecting Randomness
  - The Wrong Conclusion Ratio
- Accounting for Variability
- Conclusions

# Simulation Infrastructure

- Workloads
  - Two scientific and five commercial benchmarks
- Target System: E10000-like 16-node system
- Full System Simulation
  - Virtutech Simics running Solaris 8 on SPARC V9
  - A blocking processor model (Simics)
  - An OoO processor model (TFSim – Mauer et al., SIGMETRICS'02)
- Memory system simulator
  - MOSI invalidation-based broadcast coherence protocol (Martin et al., HPCA-02)

# Simulating Space Variability?

- Simulations are deterministic
- Variability cannot be ignored for multi-threaded applications
  - One execution may not be representative
  - Execution paths affect simulation conclusions

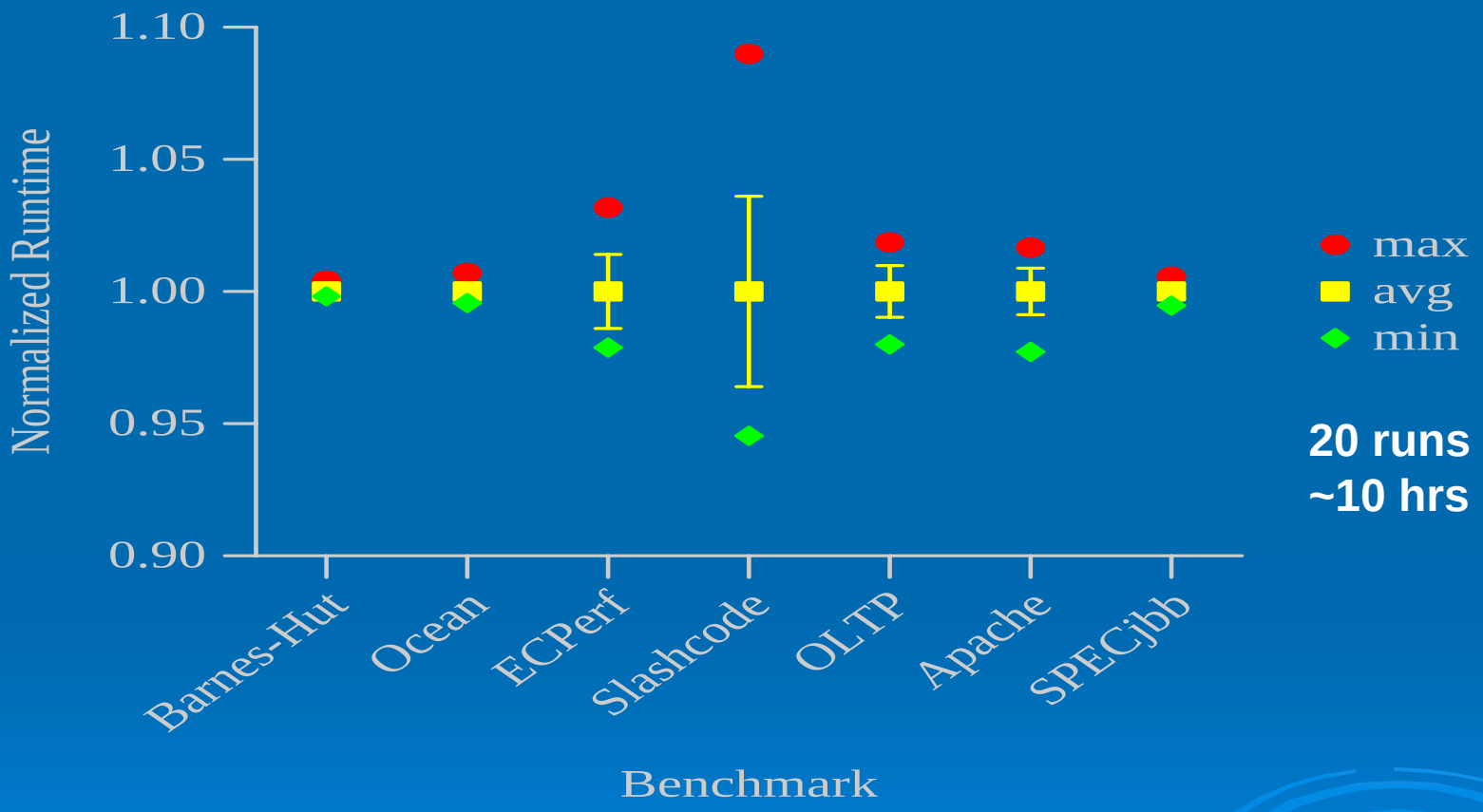
➲ We need to obtain a space of results

# Injecting Randomness

- We introduce artificial random perturbations in each simulation run
- For each memory access, latency in nanoseconds becomes Latency + r

  (r = -2, -1, 0, 1, 2 nanoseconds, uniform dist.)
- Roughly models contention due to DMA traffic
- Other methods are possible

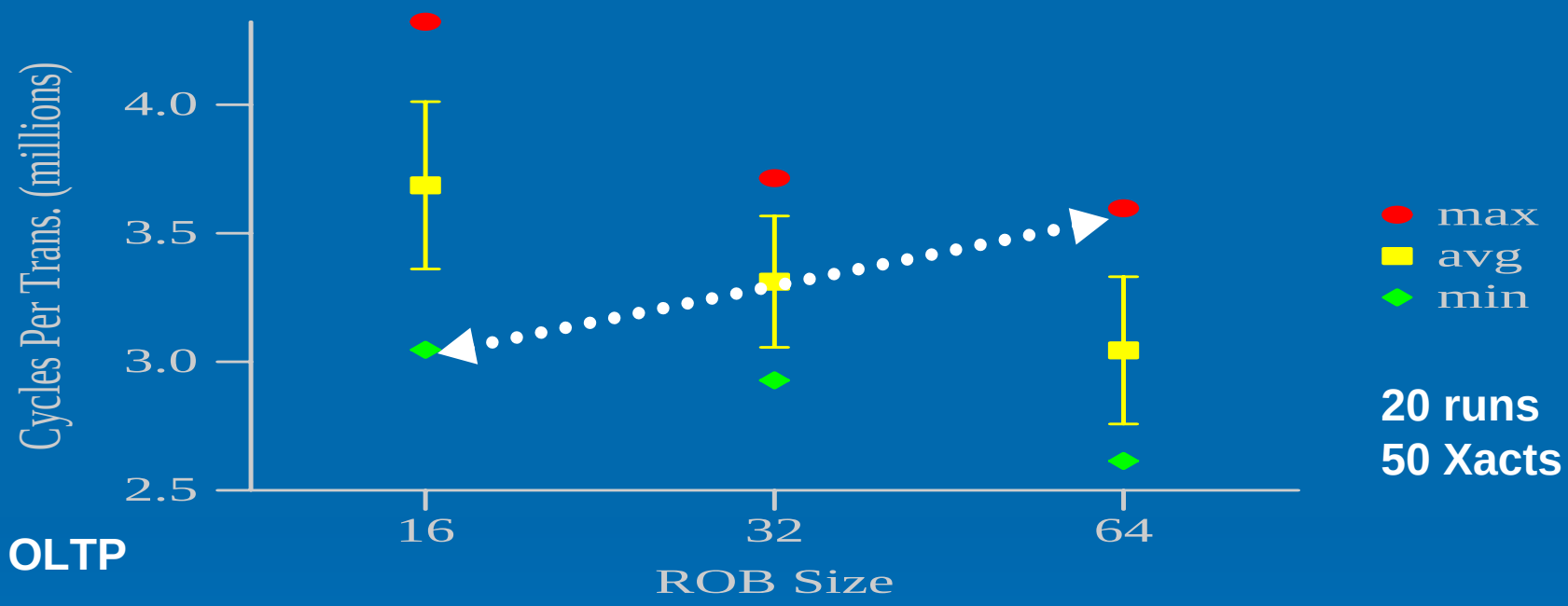
# Simulated Space Variability




**20 runs**
**~10 hrs sim.**

- Space variability exists in our benchmarks

# Quantifying Variability: The Wrong Conclusion Ratio (WCR)




- WCR (16,32) = 18%
- WCR (16,64) = 7.5%
- WCR (32,64) = 26%

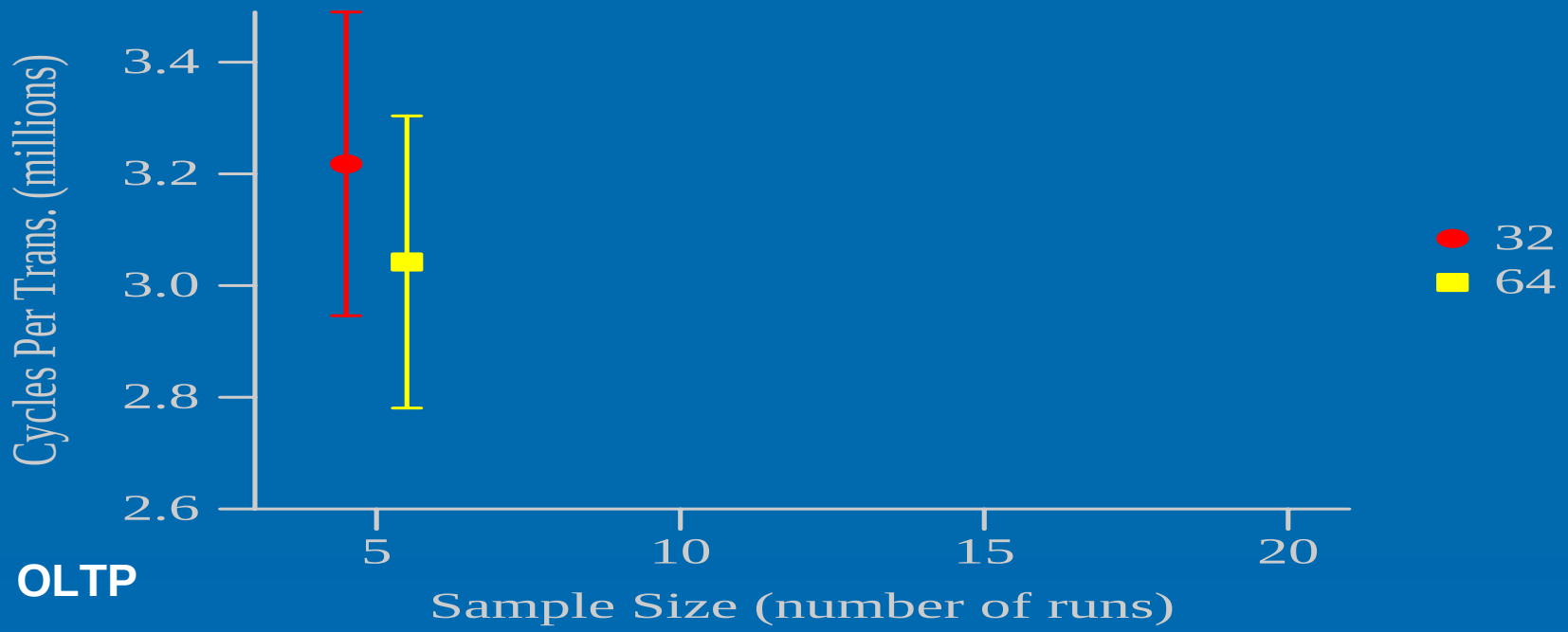
# Outline

- Motivation and Overview
- Variability in Real Systems
- Variability in Simulations
- Accounting for Variability
- Conclusions

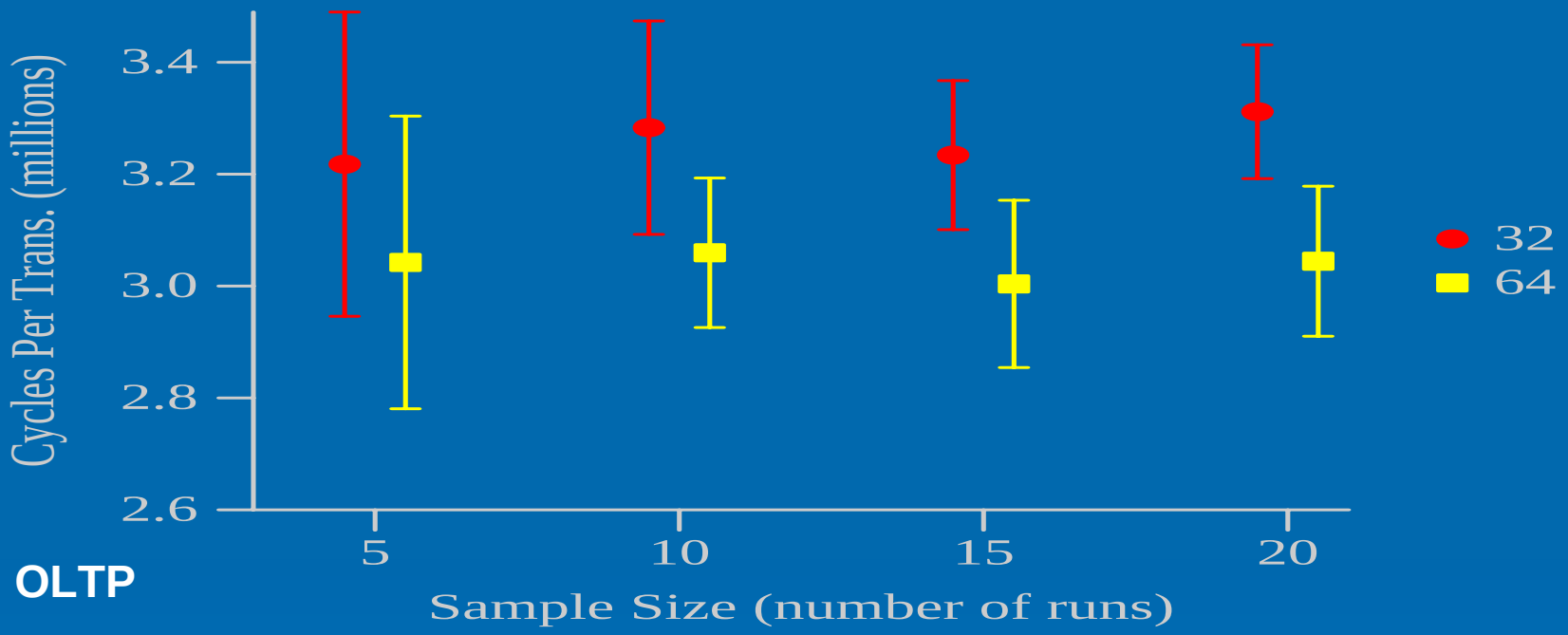
# Confidence Intervals

- Definition:
  - Range of values expected to include population parameter (e.g. mean)
- Confidence Probability:
  - Probability that true mean lies inside confidence interval
- For the same confidence probability:
  - Sample Size ↑ → Confidence Interval ↓

# Accounting for Space Variability



# Accounting for Space Variability



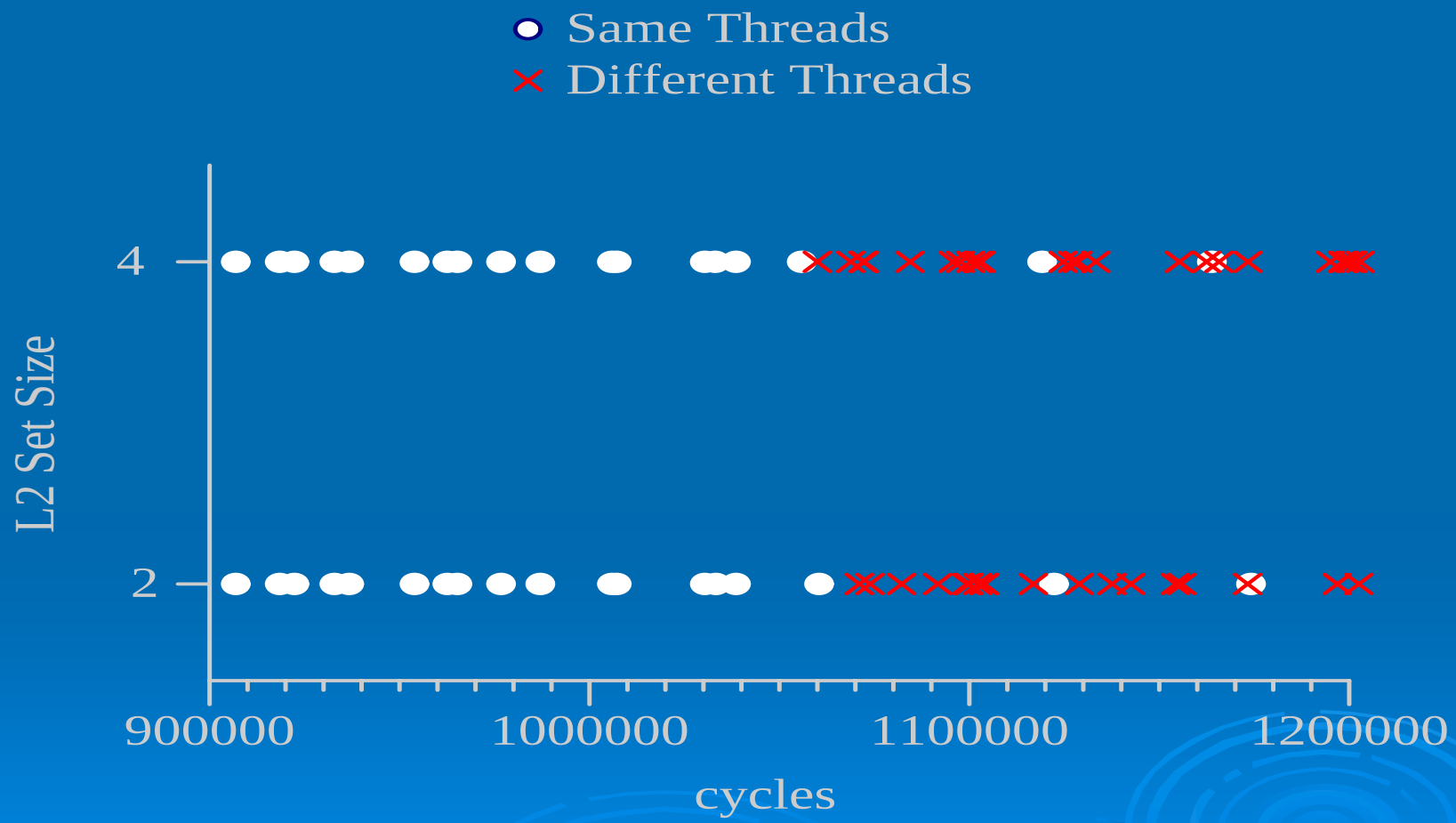

- Simple solution: Estimate #runs such that confidence intervals do not overlap
- Tests of hypotheses can be used (paper)

# Conclusions

- Short runs of multi-threaded workloads exhibit variability
- Variability can lead to wrong simulation conclusions
- Our Solution:
  - Injecting randomness
  - Multiple runs
  - Apply statistical techniques

# Backup Slides

# Effects of OS Scheduling



# WCR Definition

- Percentage of comparison simulation experiments that reach a wrong conclusion
- The correct conclusion is the relationship between averages of the two populations
- WCR can be used to estimate the wrong conclusion probability for single experiments

# Confidence Intervals - Equations

- The confidence interval for the mean of a normally distributed infinite population:

$$\bar{y} - \frac{ts}{\sqrt{n}} \leq mean \leq \bar{y} + \frac{ts}{\sqrt{n}}$$

- Sample Size needed to limit mean relative error to r:

$$n = \left( \frac{tS}{r\,\bar{Y}} \right)^2$$

# Hypothesis Testing

- Tests whether there is no difference between two population means
  - Hypothesis: $\mu_{32} = \mu_{64}$ tests whether the two means of the 32 and 64 ROB configurations are different
- Hypothesis is tested using sample means and variances
- If hypothesis rejected ➲ Our conclusion is significant

# Accounting for Time Variability

- Is time variability caused by the same effects that cause space variability?
  - Use Analysis of Variance (ANOVA)
- If time variability is caused by different effects, we need to obtain a time sample
  - Observations obtained from different starting points

# Multi-threaded Workloads and Simulation

- Multi-threaded workloads are important
  - Workloads for commercial servers
  - New architectures support multi-threading
- Performance metrics are different from traditional benchmarks
  - Throughput-oriented (transactions)
  - IPC is not appropriate (idle time!)

- Simulation Challenge: Comparing systems running multi-threaded applications

# Simulation of Multi-threaded Workloads

- Simulation is slow!
  - We cannot simulate the whole workload
- Solution:
  - Run for a fixed number of transactions
  - Measure the per-transaction runtime (cycles per transaction)
  - Use to compare different systems