SYNCHROTRACE:
SYNCHRONIZATION-AWARE ARCHITECTURE-AGNOSTIC TRACES FOR LIGHT-WEIGHT MULTICORE SIMULATION

Siddharth Nilakantan, Karthik Sangaiah, Vasil Pano, Mike Lui,
Dr. Baris Taskin, and Dr. Mark Hempstead
Department of Electrical and Computer Engineering

Drexel UNIVERSITY

Agenda

- **Introduction and Motivation**
- Multi-threaded Event Trace Capture
- Event Trace Replay Framework
- Design Space Exploration
- Optimizations

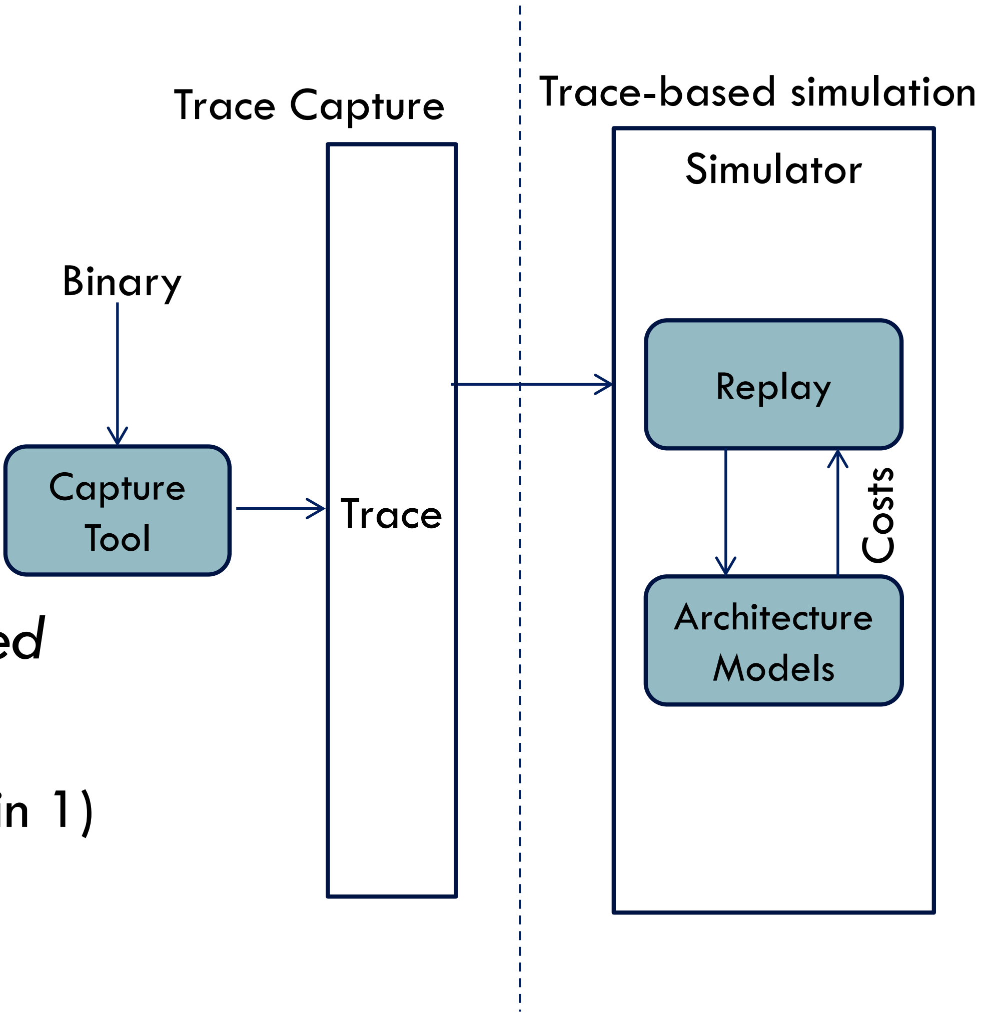
Designing Future CMPs

- Number of cores scale up ⇒ design complexity of CMPs increase
    - Memory and NoC (uncore) design space has extensive design configurations
    - Important to explore the space of the uncore as quickly as possible

- Current infrastructure in the industry
    - Execution-driven and Trace-driven simulators running multi-threaded applications
    - Execution-driven simulators
        - Simulate full instructions ⇒ **High Simulation Complexity**
        - Some examples: Gem5, Sniper, Zsim
    - Trace-driven simulators
        - Discussed next

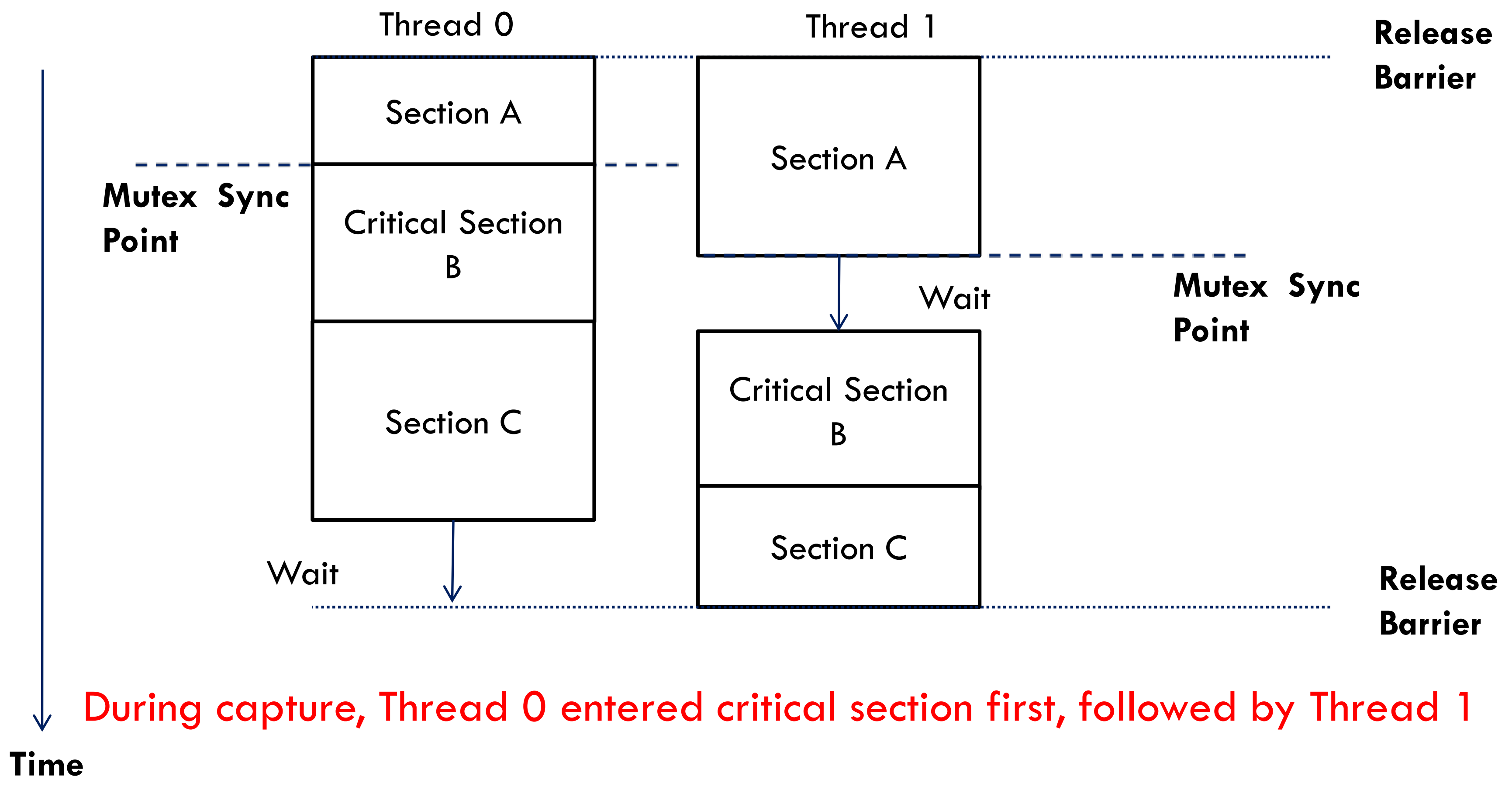
Revisiting Trace-based Simulation

- Benefits
  - Reduced simulation complexity
    - **Order of magnitude speedup!**
  - Potential scalability and Portability
- Challenge for *multi-threaded* applications:
  - Modeling non-determinism in 1) Capture and 2) Replay



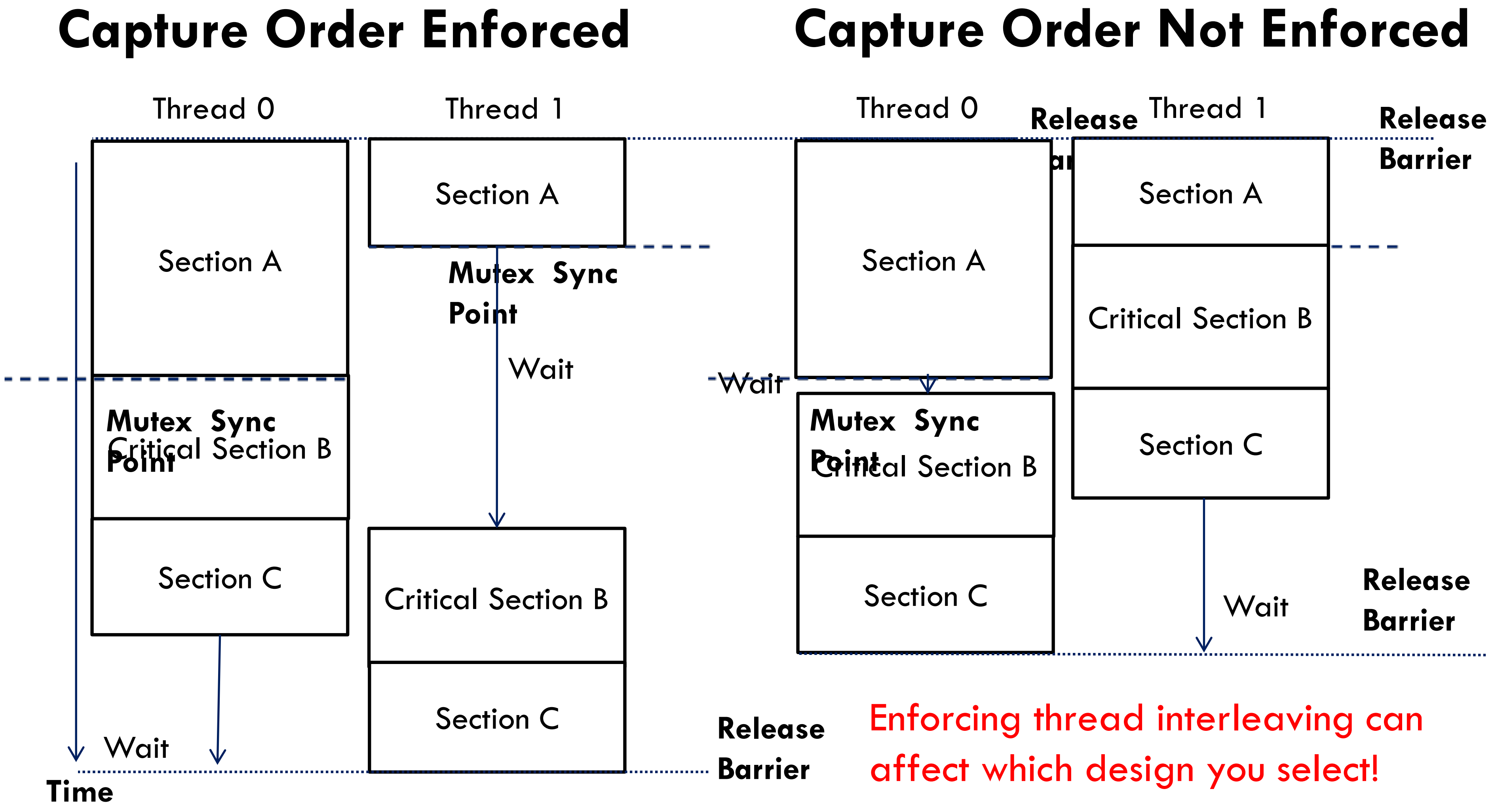
Thread Non-Determinism: Capture

Thread Interleaving at Capture



During capture, Thread 0 entered critical section first, followed by Thread 1

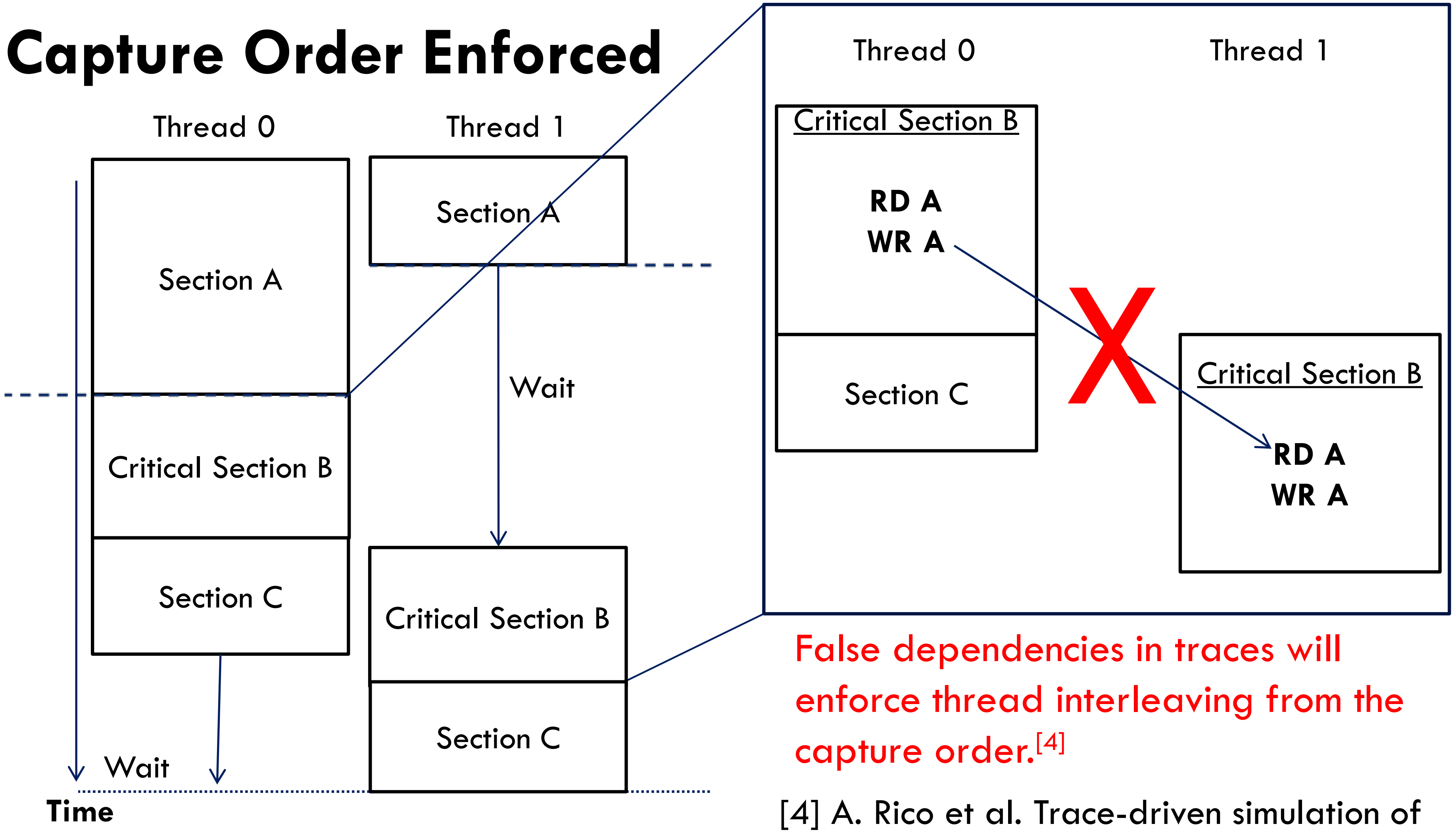
Thread Non-Determinism: Replay



Enforcing thread interleaving can affect which design you select!

Thread Non-Determinism: Cause

Capture Order Enforced



False dependencies in traces will enforce thread interleaving from the capture order.[4]

[4] A. Rico et al. Trace-driven simulation of multithreaded applications. ISPASS '11.

Prior Work in Multi-threaded Traces

- PinPlay [5]
  - Captures each thread's execution into 'Pinballs'
  - A framework for deterministic replay

- Trace-driven simulation of multithreaded applications [4]
  - Execution-driven for Threading API
  - Trace-driven for serial portions of applications
  - Complex Source-to-Source transformations

[4] A. Rico et al. Trace-driven simulation of multithreaded applications. ISPASS '11.
[5] H. Patil et al. Pinplay: A framework for deterministic replay and reproducible analysis of parallel programs. CGO '10.

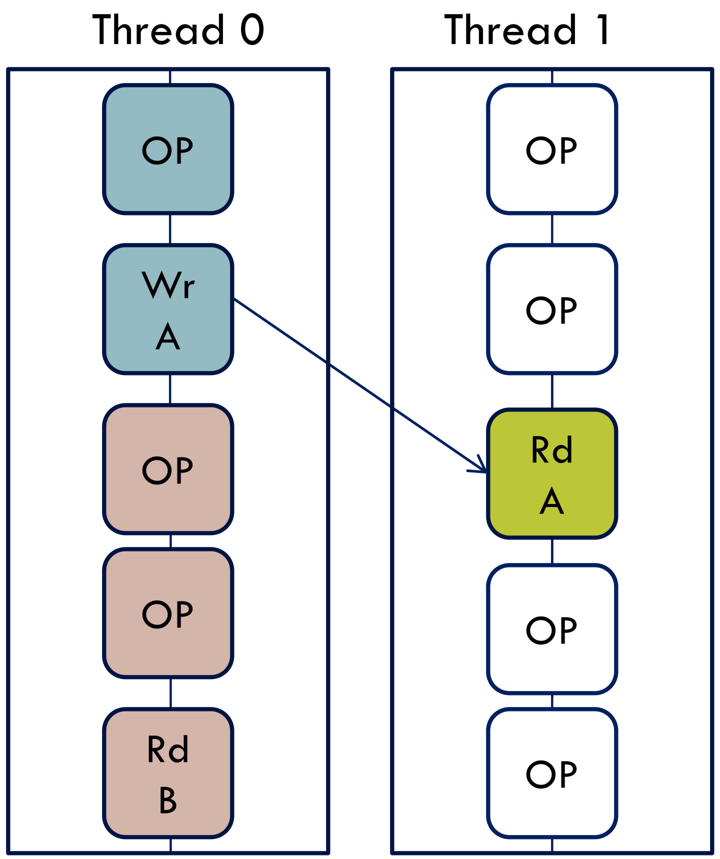
SynchroTrace

- Two Step Methodology:
  - Capture – Multi-threaded Event-Trace Capture
  - Replay – Event-Trace Replay Framework
- **Goal – Accurate, scalable design space exploration using trace-based simulation of multi-threaded applications.**

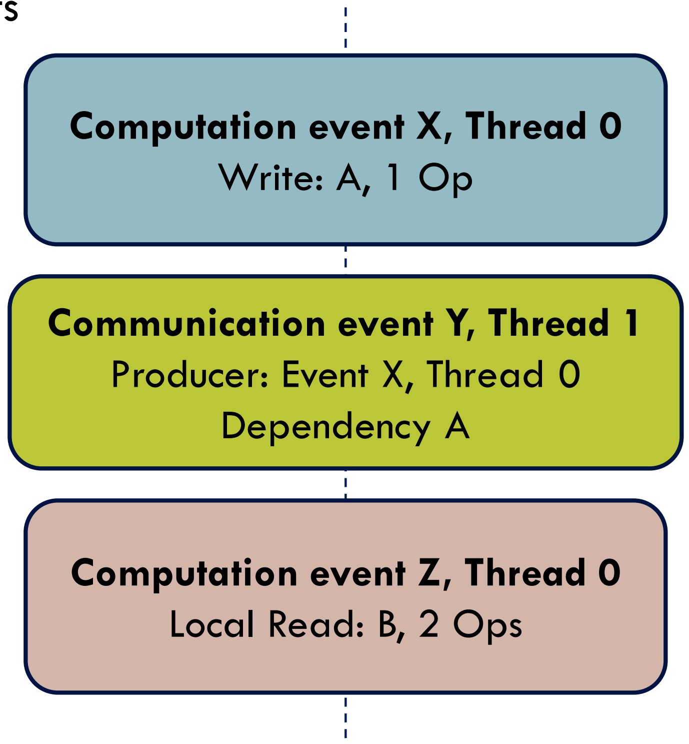
Agenda

- Introduction and Motivation
- **Multi-threaded Event Trace Capture**
- Event Trace Replay Framework
- Design Space Exploration
- Optimizations

Sigil: Thread Computation & Communication

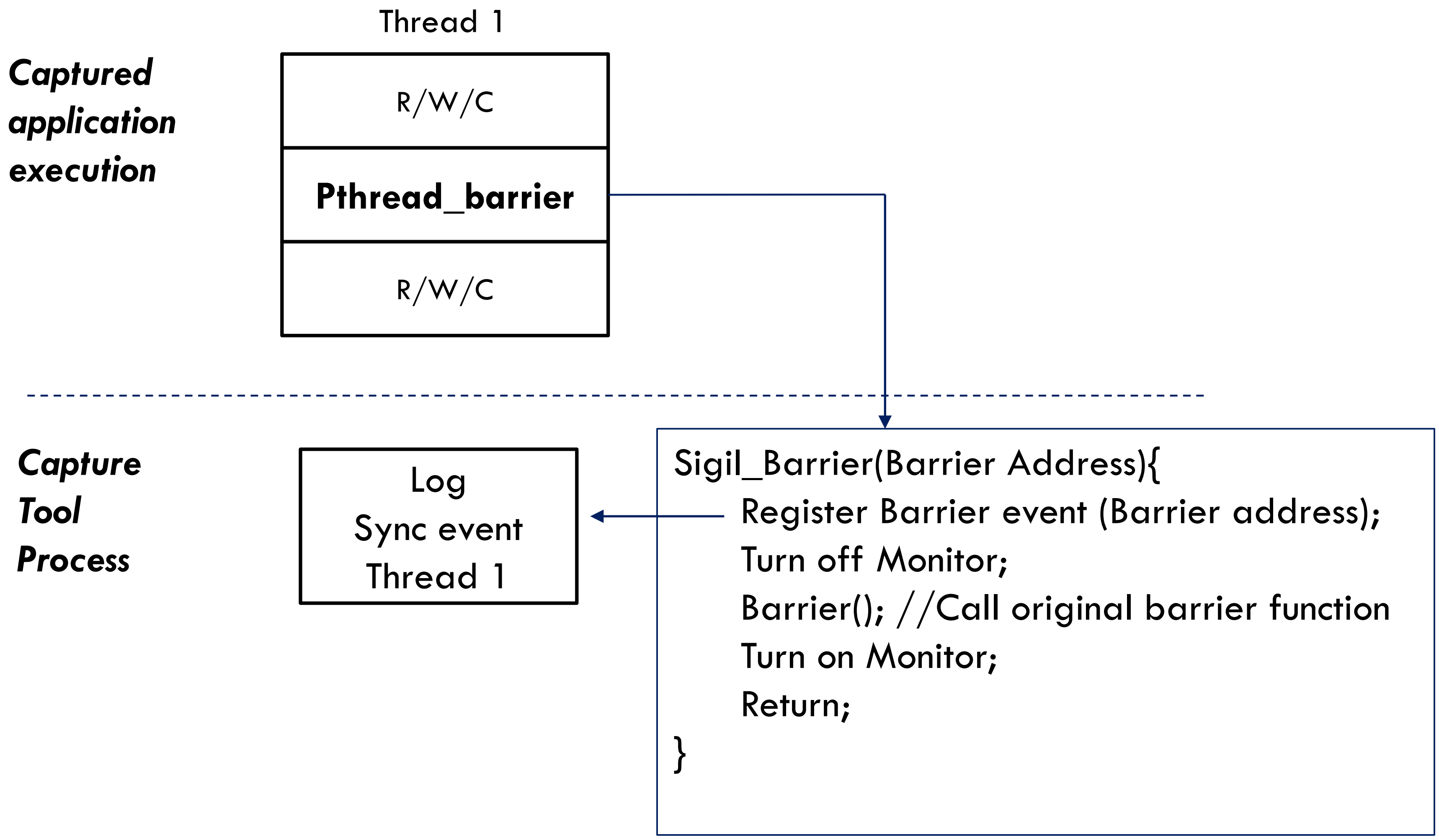


Sequence of computation and communication events


Computation event X, Thread 0
Write: A, 1 Op
Communication event Y, Thread 1
Producer: Event X, Thread 0
Dependency A
Computation event Z, Thread 0
Local Read: B, 2 Ops

Still Need Synchronization!

Intercepting Synchronization

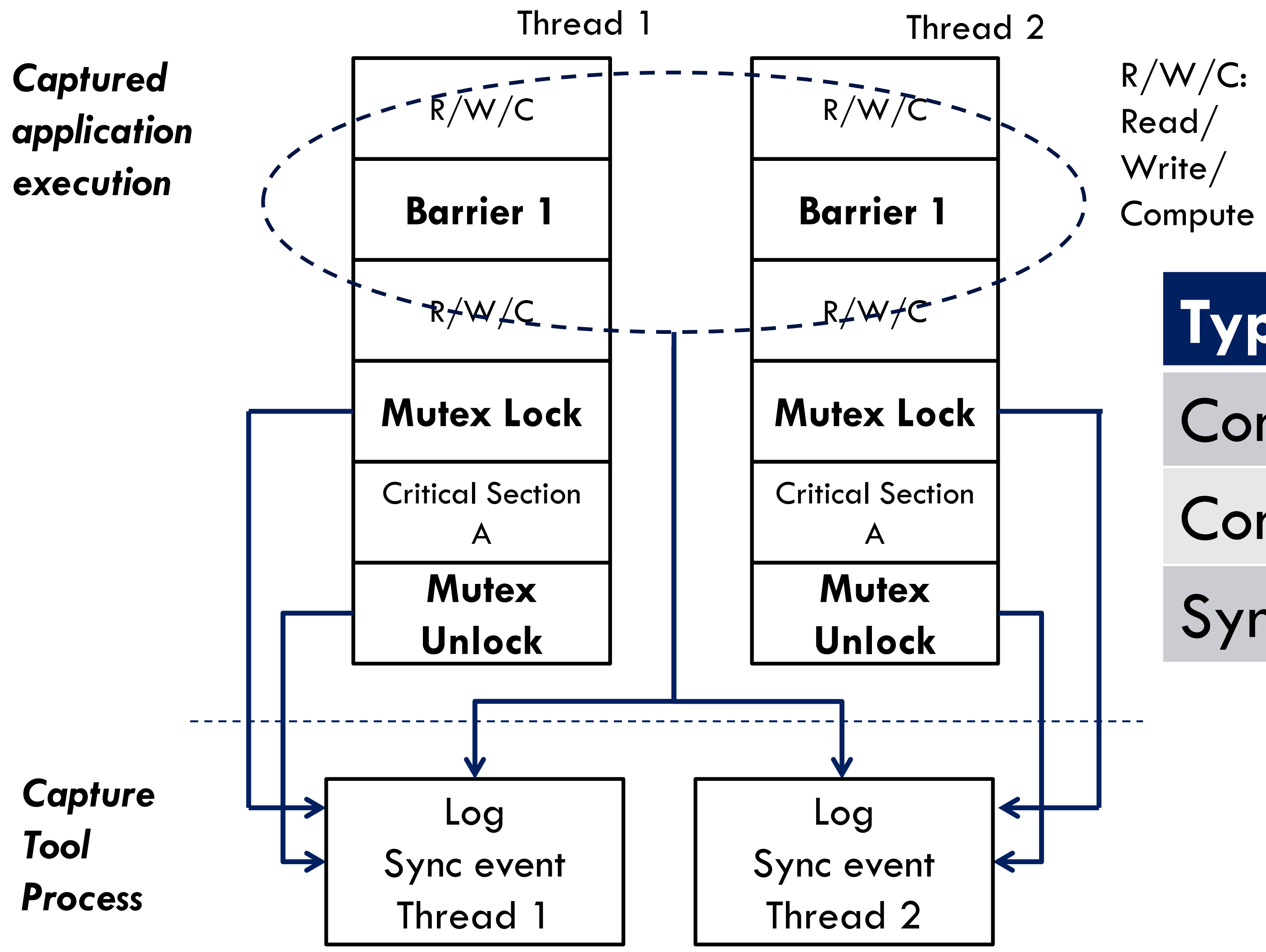


Captured application execution

Capture Tool Process

```
Sigil_Barrier(Barrier Address){
    Register Barrier event (Barrier address);
    Turn off Monitor;
    Barrier(); //Call original barrier function
    Turn on Monitor;
    Return;
}
```

Extending Event Traces to Capture Synchronization



Types of Events
Computation
Communication
Synchronization

Multi-threaded Trace Events

- Event
  - Combination of contiguous computation or communication
- Events in detail:

Listing 1: **Computation Event**

```
Event Number, Integer Op Count, Floating Point
Op Count, Memory Read Count, Memory Write Count $
Unique Addresses Written * Unique Addresses Read
```

Listing 2: **Synchronization Event**

```
Event Number, pth_ty: Pthread_Call_Type ^ Address of
Synchronization Structure
```

Listing 3: **Communication Event**

```
Event Number # Producer thread , Producer Event ,
Address Range
```

Getting SynchroTrace Capture tool

- Standalone version of Sigil with synchronization intercepts is provided @ https://github.com/dpac-vlsi/Sigil
  - Build and run instructions in above link
  - Example also provided in Sigil/example
    - FFT 8 threads, base problem size (M=16)
    - Bash script to generate traces
    - Also described on the same page

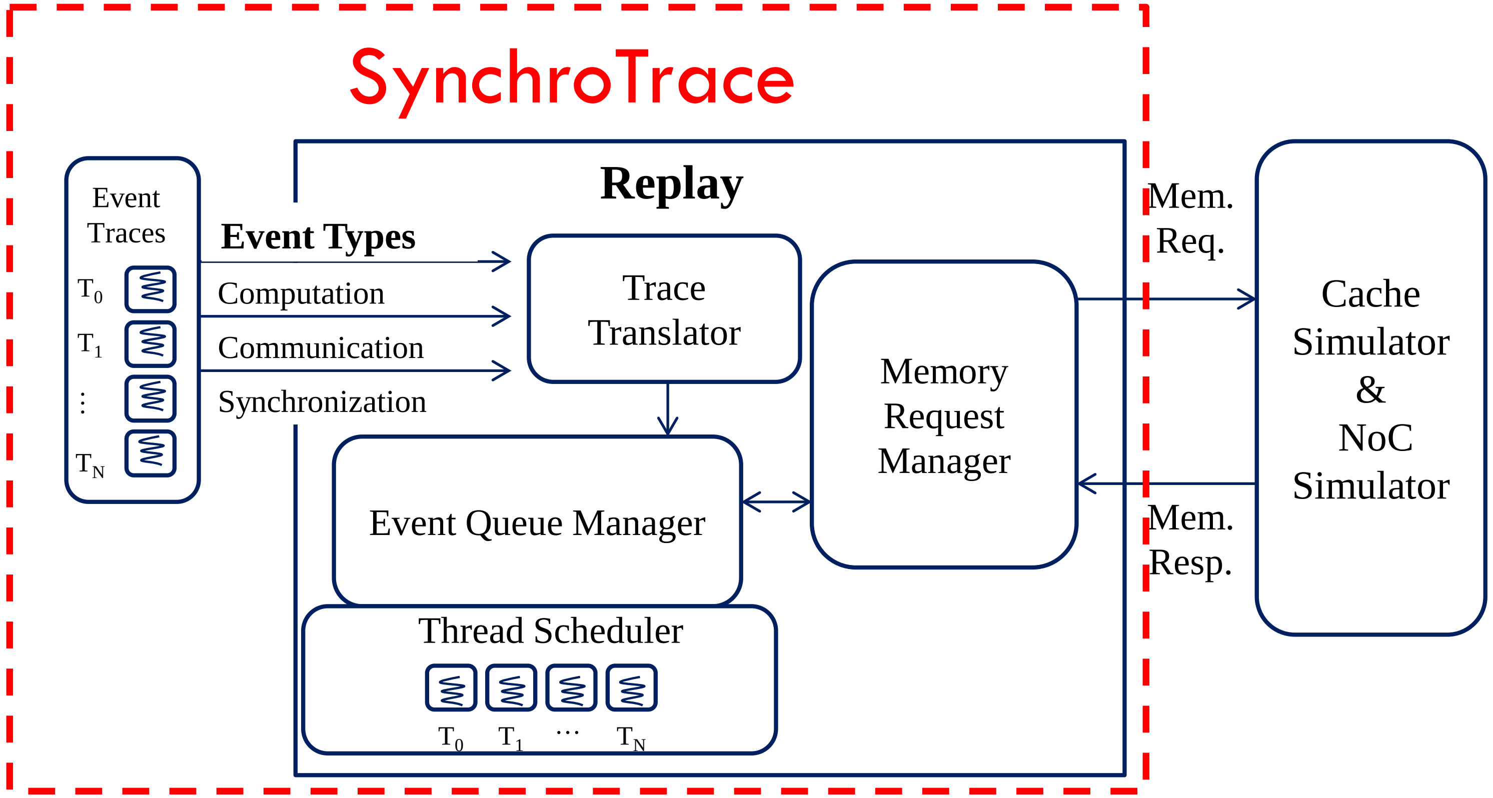
- Will go through this example during hands-on

Agenda

- Introduction and Motivation
- Multi-threaded Event Trace Capture
- **Event Trace Replay Framework**
- Design Space Exploration
- Optimizations

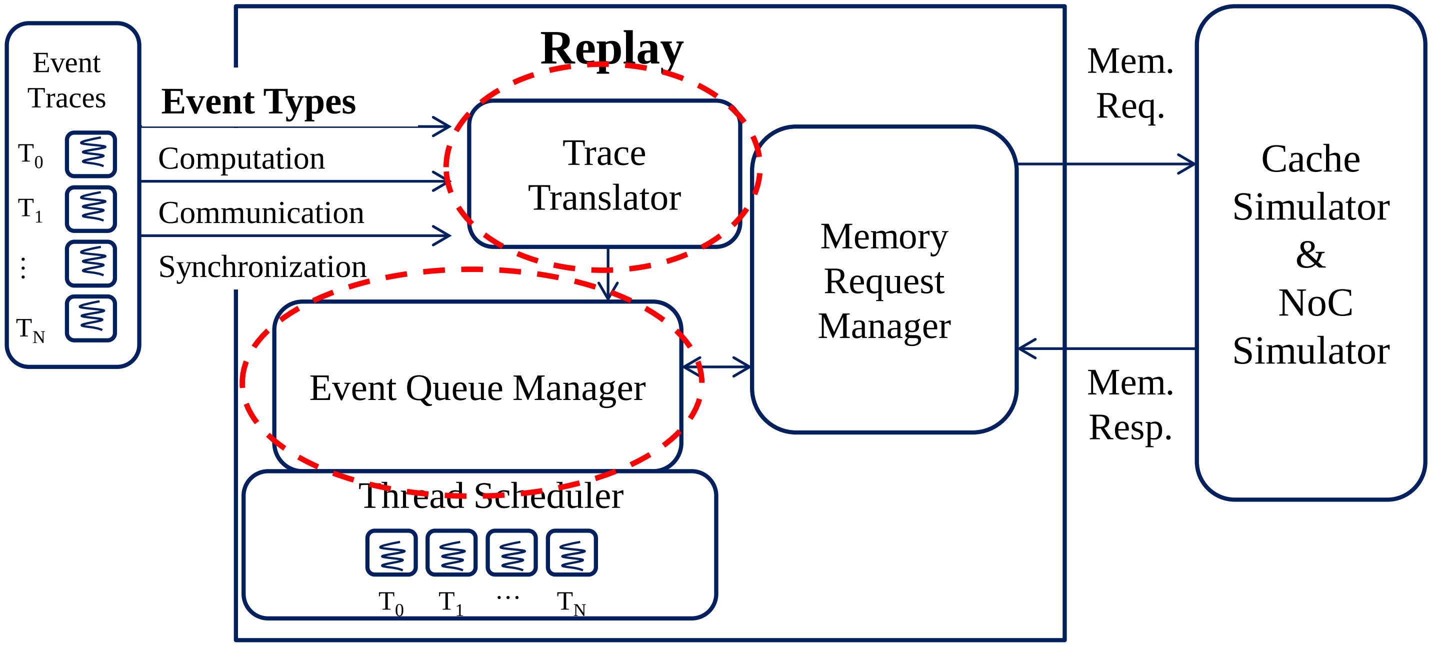
Event-Trace Replay Framework

- Traces processed in replay mechanism, which interfaces with ‘uncore’ simulators
- Prototype simulation framework integrated with Ruby and Garnet
  - 1-CPI timing model



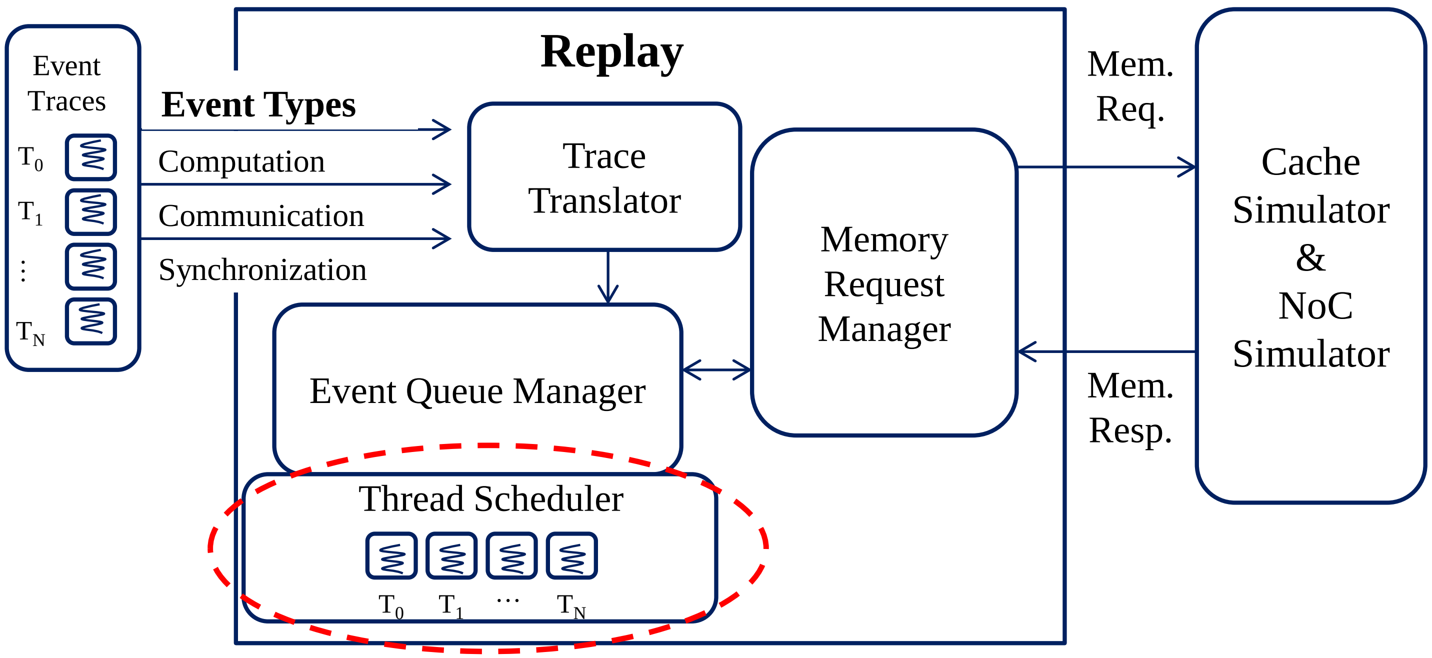
Event-Trace Replay Framework

- Trace events converted to timing events and memory requests
- Central event queue in charge of "Timing" thread progress
  - Based on compute operations and memory timing
  - Controls when to issue memory requests



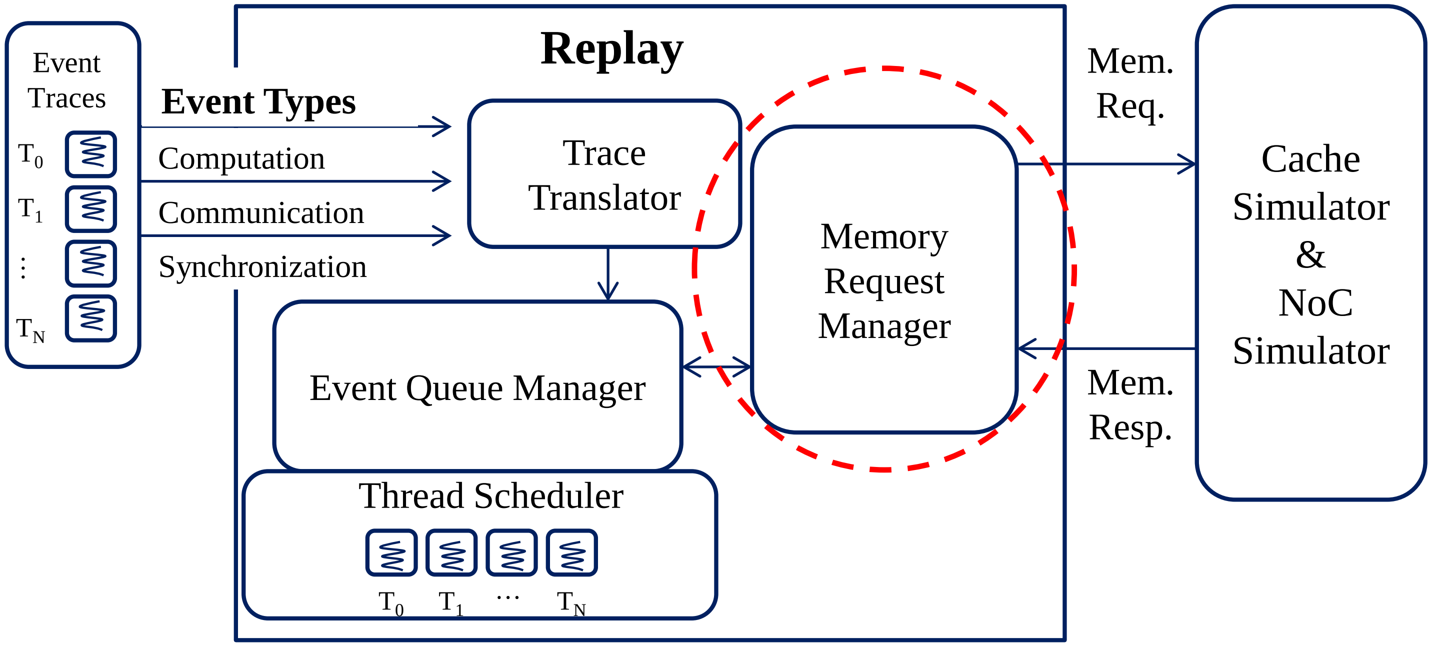
Event-Trace Replay Framework

- Thread Scheduler
  - Creates/Manages Thread State
  - Smartly swaps threads on to cores
  - Generates appropriate action for synchronization events
- Can be modified for OS thread scheduling research!



Event-Trace Replay Framework

- Memory Request Manager
  - Interface to Cache and NoC simulators



Replay Integration with simulator

- Recall: Replay code must be integrated with uncore models
- Current Replay hooked up to Gem5's Cache and NoC simulators (Ruby + Garnet)
  - Most tested SynchroTrace release is with Gem5 version from April 2013
  - Newer release with latest Gem5 in testing with support for Gem5's classic memory
- WIP: Replay integrated with other simulators

Getting Replay

- Get the Replay code @ https://github.com/dpac-vlsi/SynchroTrace
  - Integrated with Gem5 – Standalone copy
  - Build and run instructions provided in above link
- Building
  - Ensure system is configured appropriately to run gem5. Info on gem5 configuration is available at http://gem5.org/Dependencies
- Will go through build and run example during hands-on

Running Replay

- Running – steps on Github
  - Sample traces from Sigil run are provided
    - FFT 8 threads, base problem size (M=16)
  - Example can also be run
    - Perl script to run: run_synchrotrace_fft.pl
- Gem5 usually run with a python configuration script
  - We provide SynchroTrace configuration script
    - Script connects the Replay code to Ruby
  - **Can modify configuration script** to avail of plethora of gem5 cache and NoC models!
    - Visit http://www.gem5.org/Ruby

Agenda

- Introduction and Motivation
- Multi-threaded Event Trace Capture
- Event Trace Replay Framework
- **Design Space Exploration**
- Optimizations

Memory/NoC Design Space Exploration

- Goal:
  - **Is trace-based simulation of multi-threaded applications a viable solution for design space exploration?**
- *Lets compare to Gem5!*

- Methodology:
  1. Apply area constraints
     - 75%, 33% of **max** area in subset design space
  2. Apply equivalent Power constraints on SynchroTrace and Gem5
     - 75%, 33% of **max** power in subset design space
  3. Find the highest performing design point within constraints and compare between frameworks

Memory/NoC Design Space Exploration

- 8-Core CMP
- 8-Threaded Splash2 and PARSEC benchmarks
- 16 Total Design Configurations

TABLE I: Cache Design Parameters

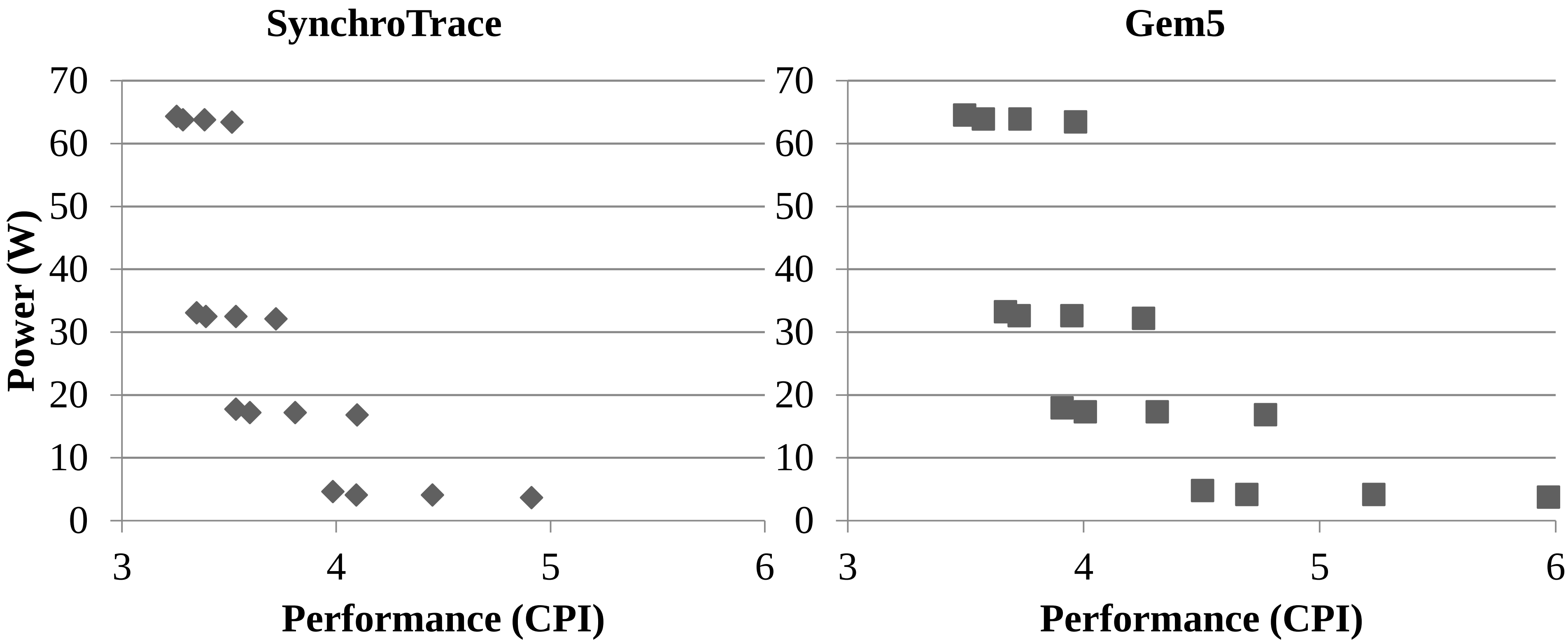
Cache Configurations	Cache Sizes
SS (Small)	L1 I/D = 4kB; L2 = 256kB
MM (Medium)	L1 I/D = 16kB; L2 = 1024kB
LL (Large)	L1 I/D = 32kB; L2 = 2048kB
vLvL (Very Large)	L1 I/D = 64kB; L2 = 4096kB

TABLE II: NoC Design Parameters

Network Configurations	Network Parameters
VC_2	Virtual Channels = 2 Buffer Depth = 4
VC_4	Virtual Channels = 4 Buffer Depth = 4
BW_4	Link Bandwidth = 4 Bytes
BW_16	Link Bandwidth = 16 Bytes

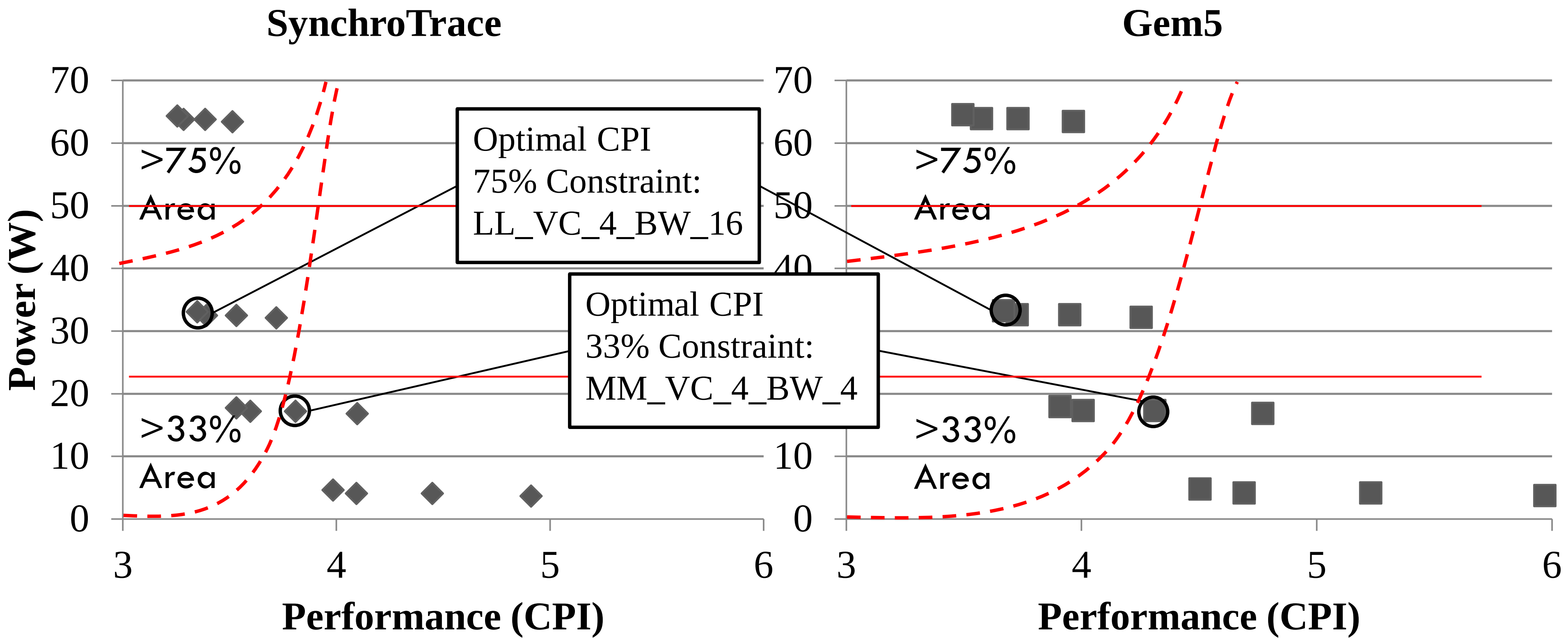
Uncore Power vs. Performance

- SynchroTrace and Gem5 Comparison
  - Trends match
  - Relative CPIs between design points within 97%



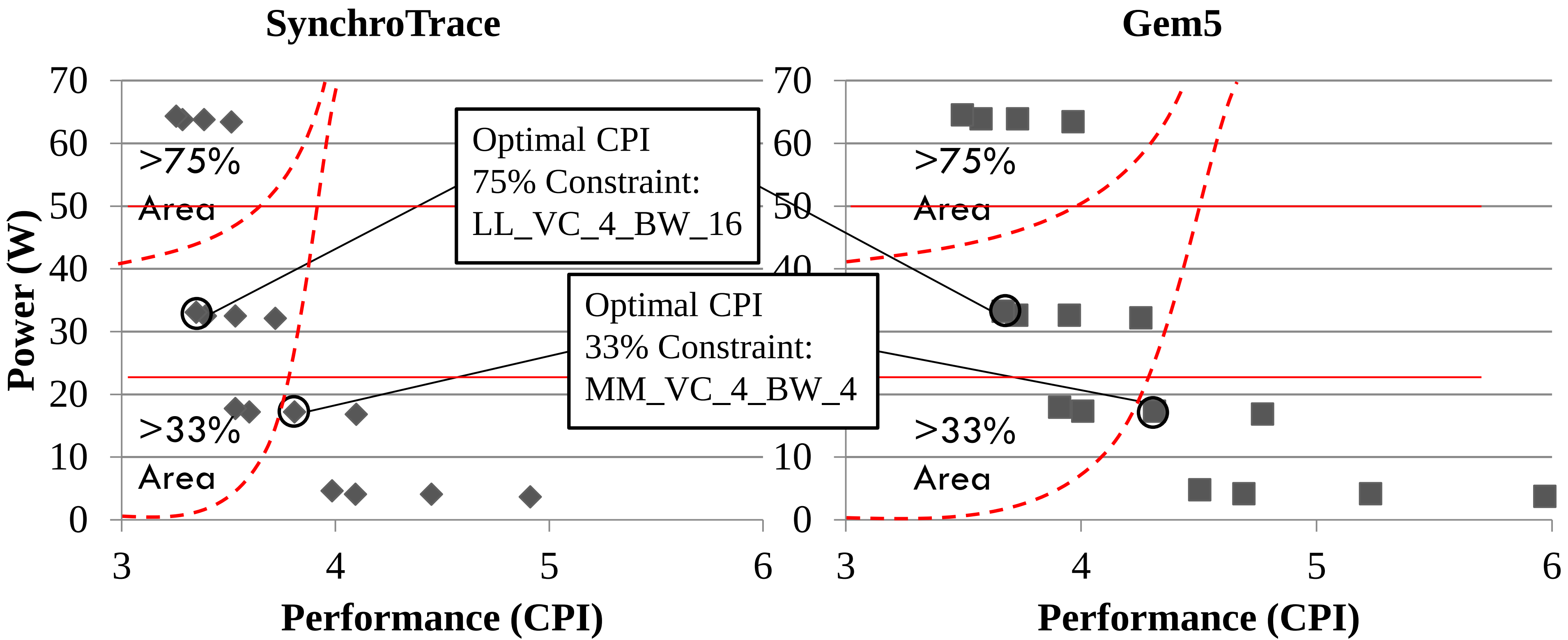
Uncore Power vs. Performance

- Apply Area Constraints (75% and 33%) and Power Constraints



Uncore Power vs. Performance

- **Accuracy** - Identical design points selected
- **Speedup** - SynchroTrace up to 13.4x faster than Gem5

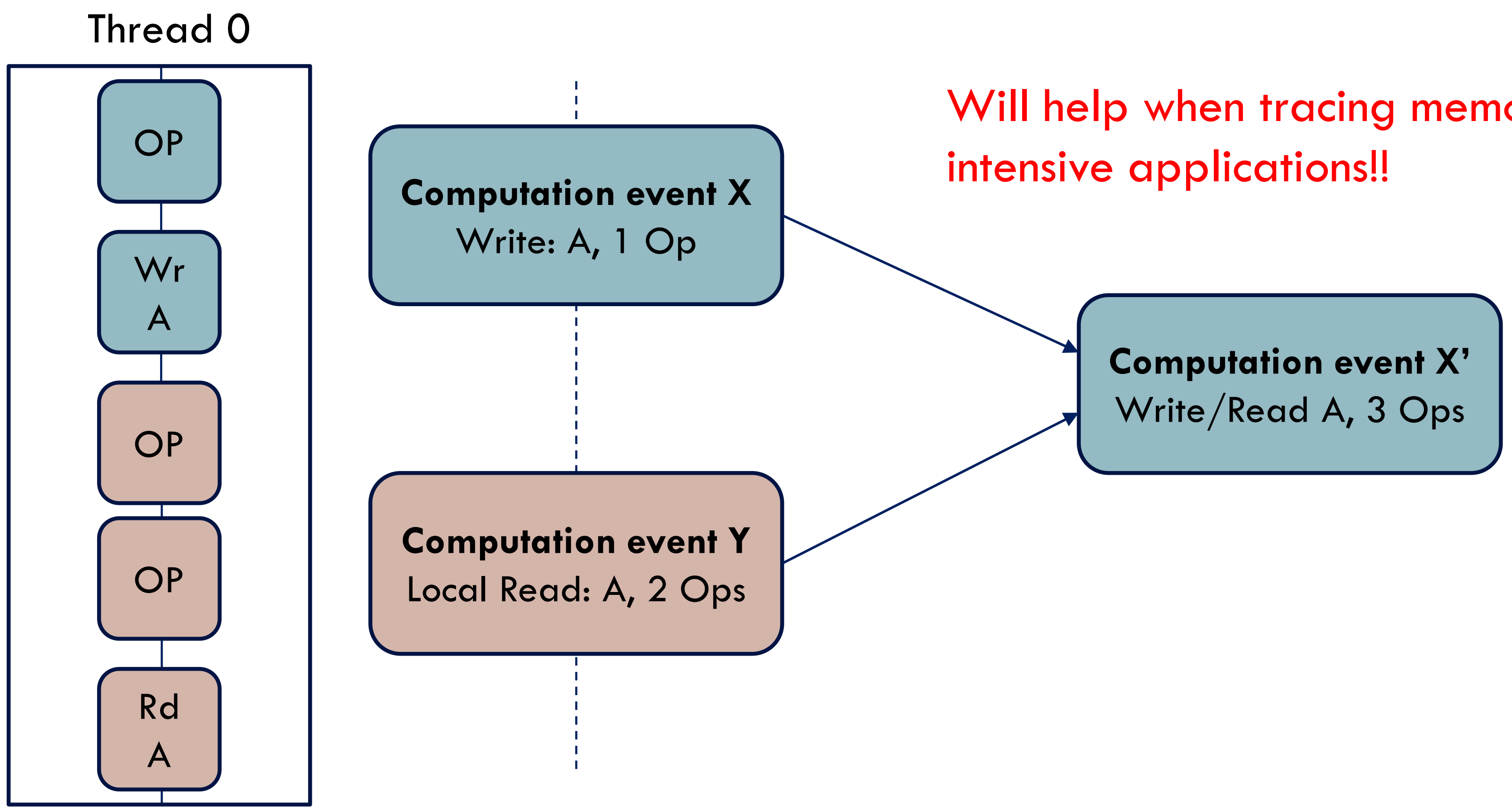


Agenda

- Introduction and Motivation
- Multi-threaded Event Trace Capture
- Event Trace Replay Framework
- Design Space Exploration
- **Optimizations and Conclusions**

Optimization 1 – Trace Compression

- Merge Events together (up to 74% reduction in size)
- Estimate cycles for cache hits after a miss in Replay

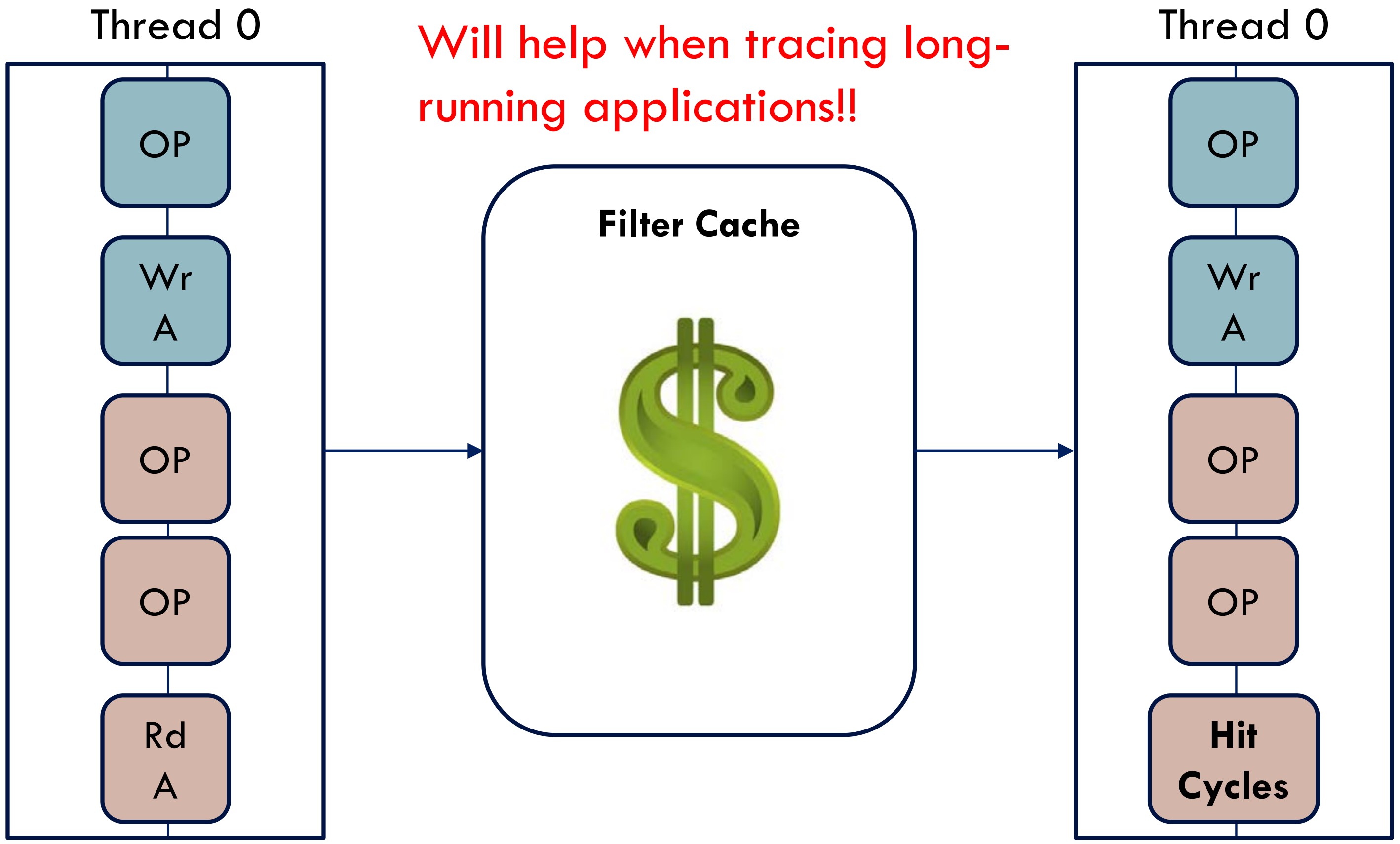


Will help when tracing memory-intensive applications!!

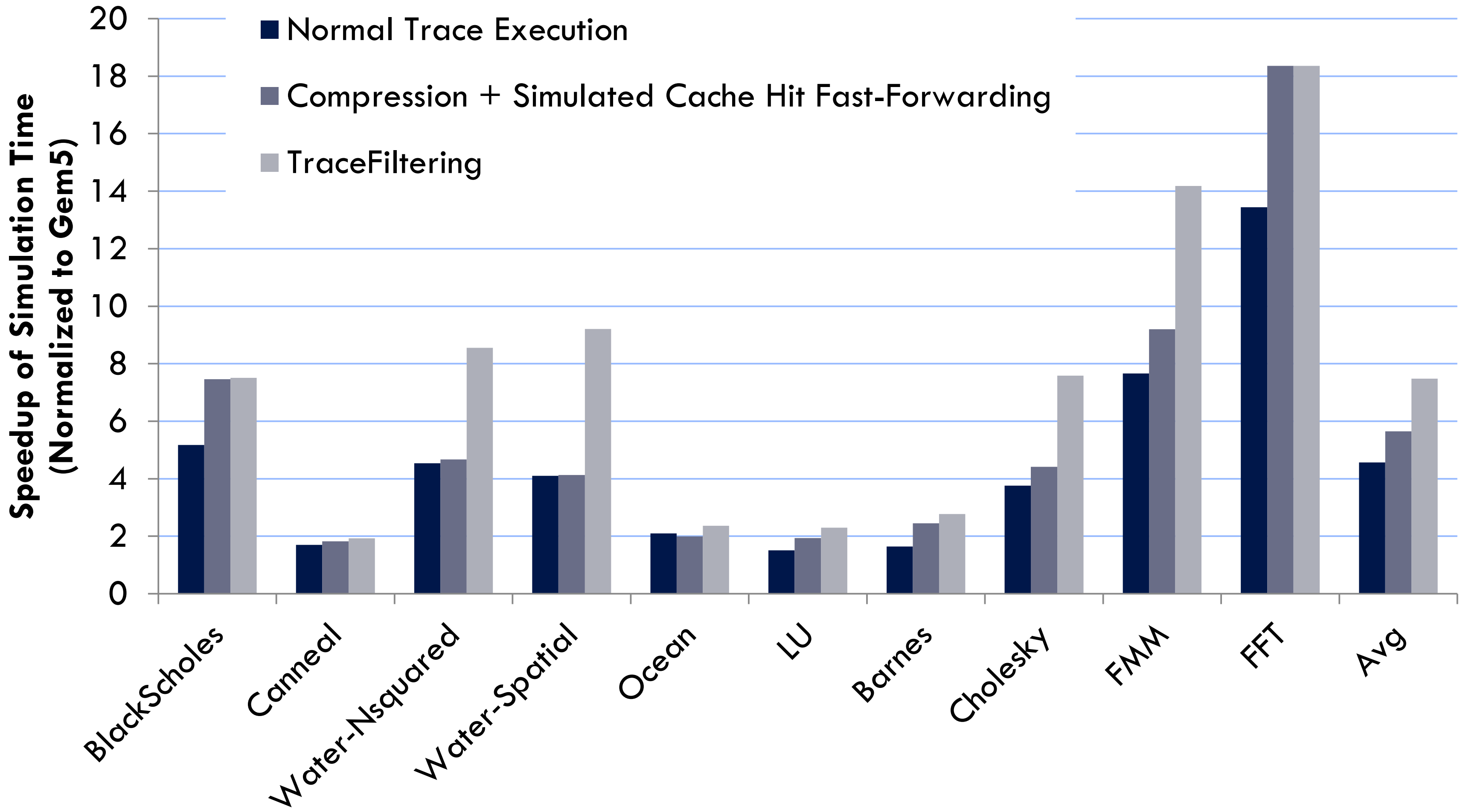
Optimization 2 – Trace-Filtering

- Post-process trace with Filter Cache
  - Filter local cache hits within trace with an 8kB fully associative filter cache

Will help when tracing long-running applications!!



Optimizations – Speedup



Conclusions

- Non-determinism – Essential to model for multi-threaded applications in trace-based simulation
  - Extended Sigil to capture Synchronization
    - Traces for Multi-threaded Applications
  - Created a prototype multi-threaded trace-based simulator
    - Replay mechanism to interface the traces to architecture models
- Advantages of SynchroTrace
  - Advantages of Trace-based approach – Portability, Speedup, Scalability
    - *But for multi-core era!*
  - Used for efficient and accurate design space exploration
  - Can leverage trace-based optimizations to improve simulation speed

Re: Getting SynchroTrace

- Available at https://github.com/dpac-vlsi
  - /Sigil
  - /SynchroTrace
- Simple build and run instructions
  - Examples for a starting point
- Modify and configure to explore a variety of designs!
- Questions? Contact
  - sid.nil@gmail.com
  - ks499@drexel.edu

References

1. N. Binkert et al. The gem5 simulator. In The ACM SIGARCH Computer Architecture Newsletter, August 2011.
2. T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In International Conference for High Performance Computing, Networking, Storage and Analysis (SC), November 2011.
3. D. Sanchez and C. Kozyrakis. Zsim: Fast and accurate microarchitectural simulation of thousand-core systems. In Proceedings of the International Symposium on Computer Architecture, ISCA '13, 2013.
4. H. Patil, C. Pereira, M. Stallcup, G. Lueck, and J. Cownie. Pinplay: A framework for deterministic replay and reproducible analysis of parallel programs. In Proceedings of the IEEE/ACM International Symposium on Code Generation and Optimization, CGO '10, 2010.
5. A. Rico, A. Duran, F. Cabarcas, Y. Etison, A. Ramirez, and M. Valero. Trace-driven simulation of multithreaded applications. In IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2011.
6. S. Nilakantan and M. Hempstead. Platform-independent analysis of function-level communication in workloads. In 2013 IEEE International Symposium on Workload Characterization (IISWC), Sept 2013.

Questions?