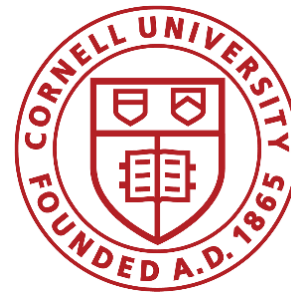


# Samsara: Efficient Deterministic Replay in Multiprocessor Environments with Hardware Virtualization Extensions

Shiru Ren, Le Tan, Chunqi Li, Zhen Xiao, and Weijia Song



# Table of Contents

**1**

**Introduction**

**2**

**Background & Motivation**

**3**

**Record & Replay Memory  
Interleaving with HAV**

**4**

**Samsara Overview**

**5**

**Evaluation**

**6**

**Conclusion**



# Introduction

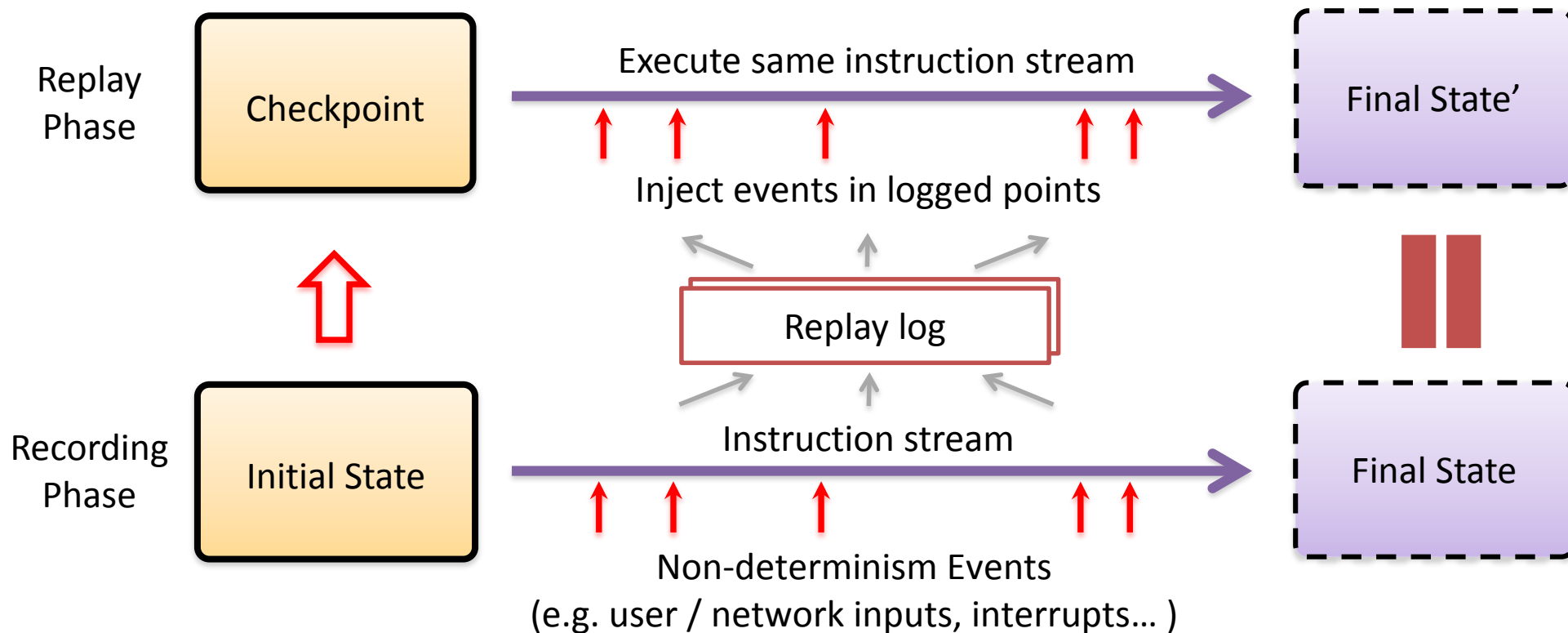
## **Nondeterminism**

- Multiprocessor architectures are inherently nondeterministic
- The lack of reproducibility complicates software debugging, security analysis, and fault tolerance

# Introduction

## Deterministic Replay

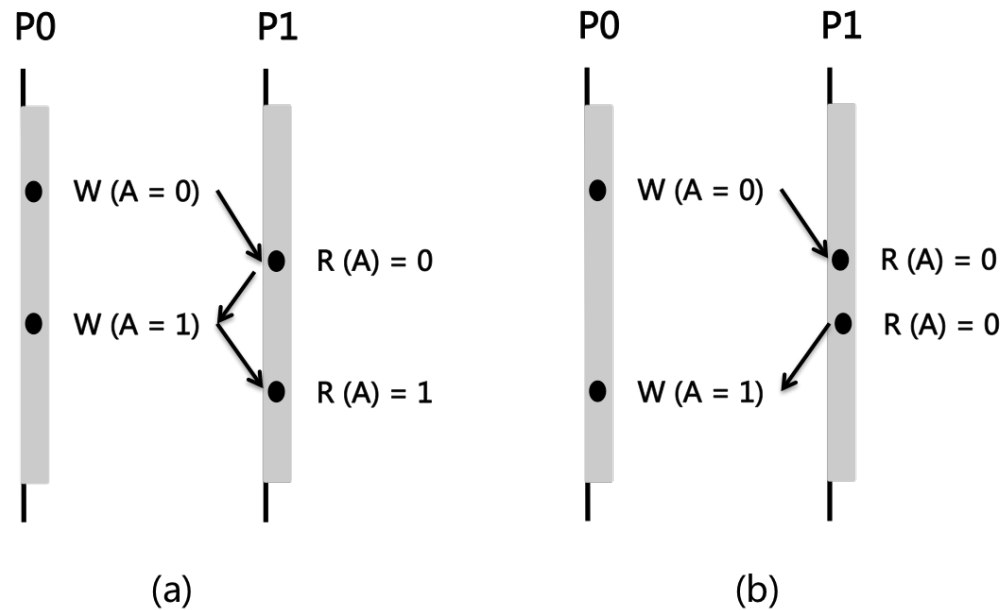
- Gives computer users the ability to travel backward in time, recreating past states and events
- Checkpoint + record all non-deterministic events



# Introduction

## Deterministic Replay for Multi-processor

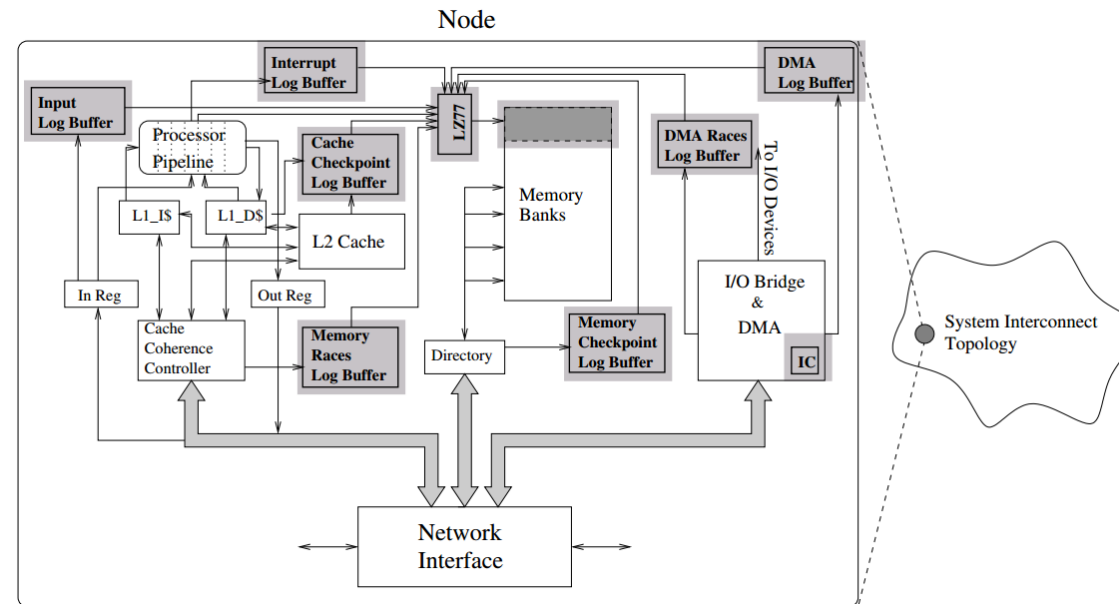
- Deterministic replay for single processor is relatively mature and well-developed
- Challenge on the multi-processor systems: **Memory Access Interleaving**



# Background & Motivation

## Hardware-based schemes

- Use special hardware support for recording memory access interleaving
- Redesign the cache coherence protocol



The FDR System [ISCA '03]

# Background & Motivation

## Hardware-based schemes

- Use special hardware support for recording memory access interleaving
- Redesign the cache coherence protocol

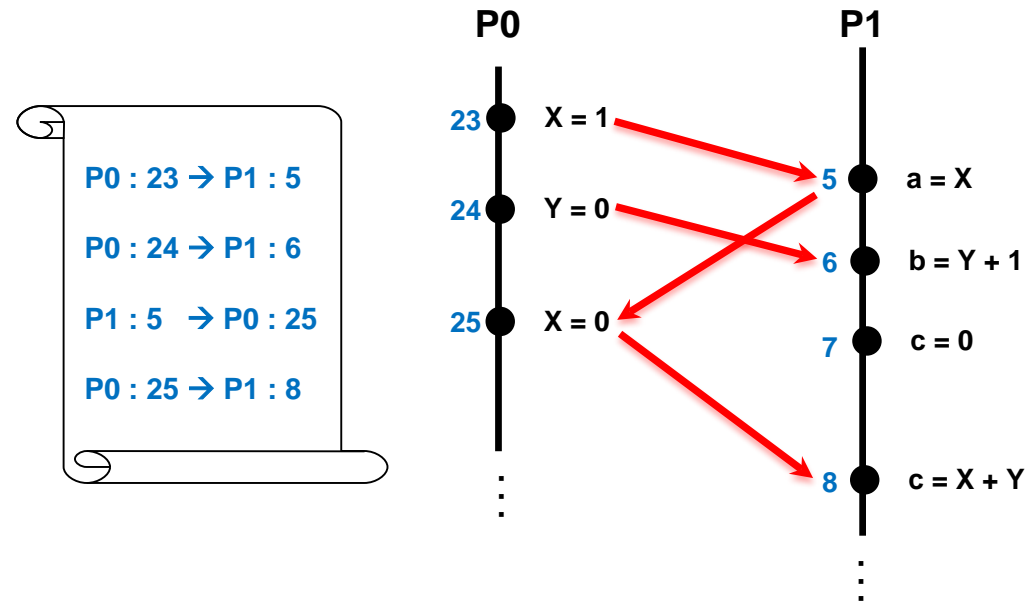
## Issues

- Increases the complexity of the circuits, impractical for use in real systems
- Huge space overhead which limits the duration of the recorded interval

# Background & Motivation

## Software-only schemes

- Modify OS, compiler, runtime libraries or VMM
- Virtualization-based approaches—CREW protocol
- CREW: Concurrent-Read & Exclusive-Write





# Background & Motivation

## Software-only schemes

- Modify OS, compiler, runtime libraries or VMM
- Virtualization-based approaches—CREW protocol
- CREW: Concurrent-Read & Exclusive-Write

## Issues

- Each memory access operation must be checked for logging
- Serious performance degradation (about **10x** compared to the native execution)
- Huge log size (approximately **1 MB/processor/second**)

# Background & Motivation

## To summarize

- Software-only schemes are inefficient without proper hardware support
- No commodity processor with dedicated hardware-based record and replay capability

# Background & Motivation

## To summarize

- Software-only schemes are inefficient without proper hardware support
- No commodity processor with dedicated hardware-based record and replay capability

## Our goal

- To implement a software approach that can take full advantages of the latest hardware features in commodity processors to record and replay memory access interleaving efficiently without introducing any hardware modifications.
- Hardware-assisted virtualization (HAV)  
(e.g., Intel<sup>®</sup> Virtualization Technology)





# Record & Replay Memory Interleaving with HAV

## Point-to-point logging approach (CREW protocol)

- Record dependences between pairs of instructions → Huge logs
- Large number of memory access detections (VM exit) → Excessive overhead

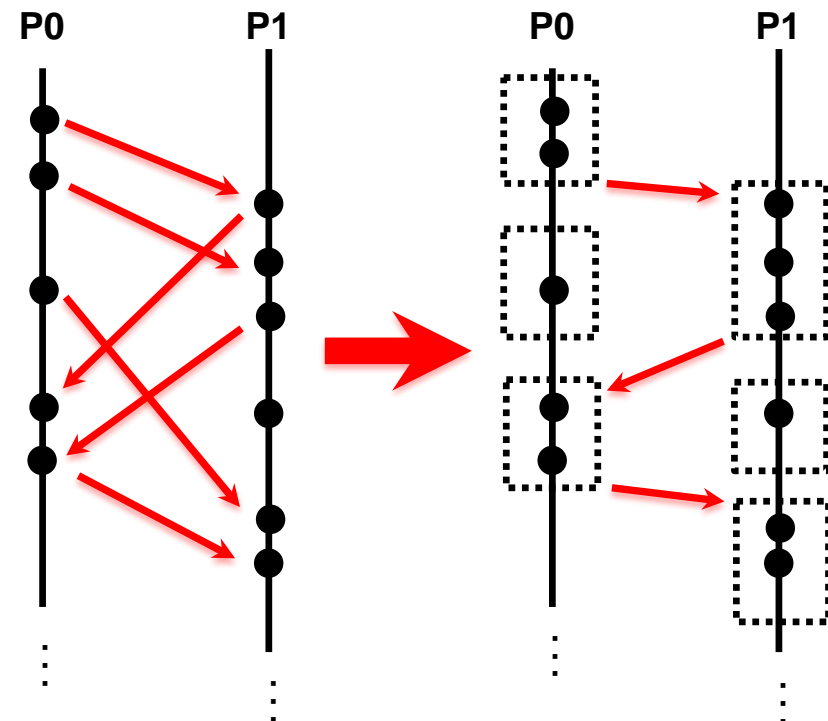
# Record & Replay Memory Interleaving with HAV

## Point-to-point logging approach (CREW protocol)

- Record dependences between pairs of instructions  Huge logs
- Large number of memory access detections (VM exit)  Excessive overhead

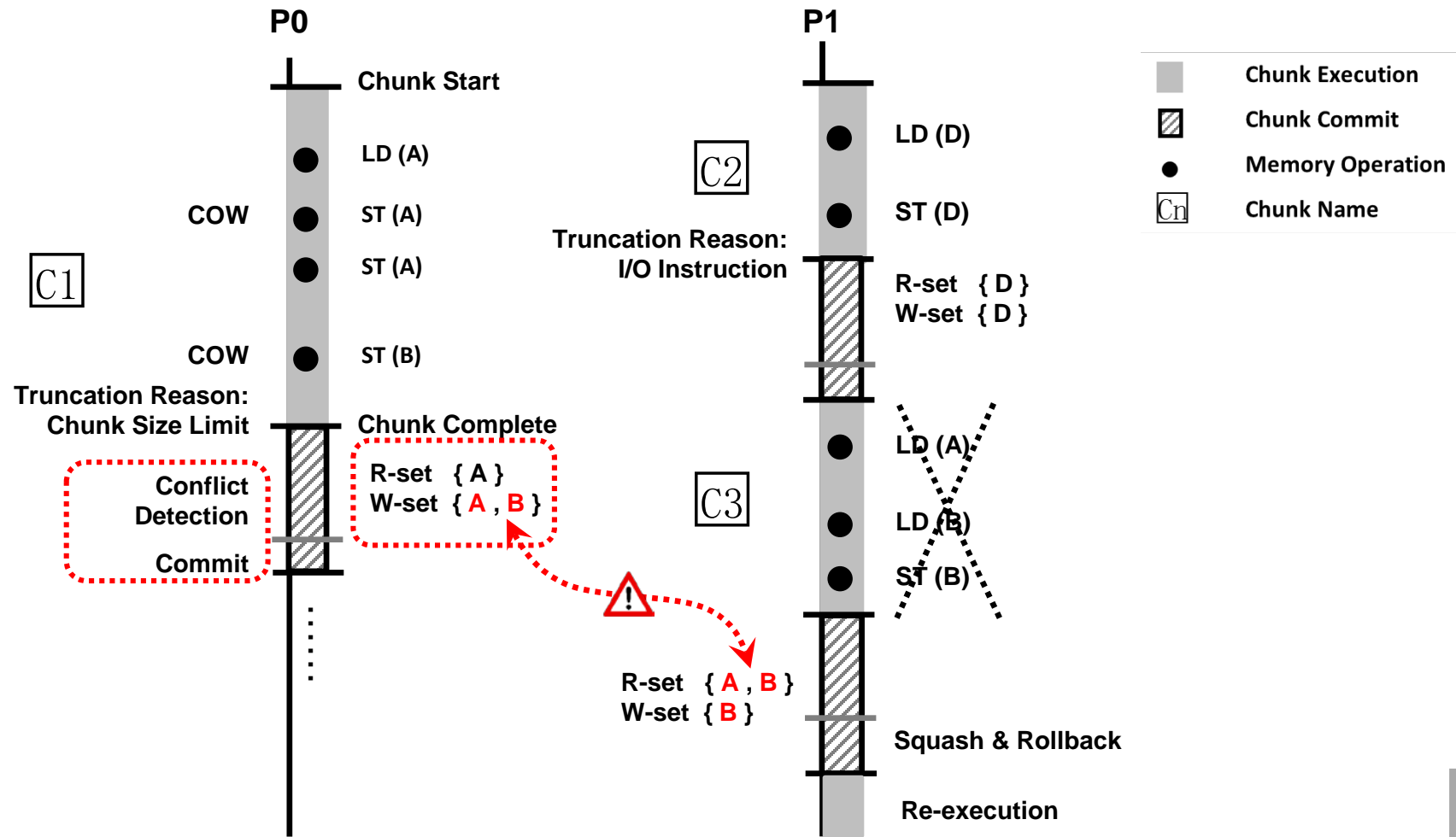
## Chunk-based Strategy

- Restrict processors' execution into a series of chunks
- Record chunk size & commit order
- Chunk execution must satisfy:
  - Atomicity
  - Serializability



# Record & Replay Memory Interleaving with HAV

- Serializability: Conflict detection, Chunk commit
- Atomicity: Copy-on-write (COW), Rollback



# Record & Replay Memory Interleaving with HAV

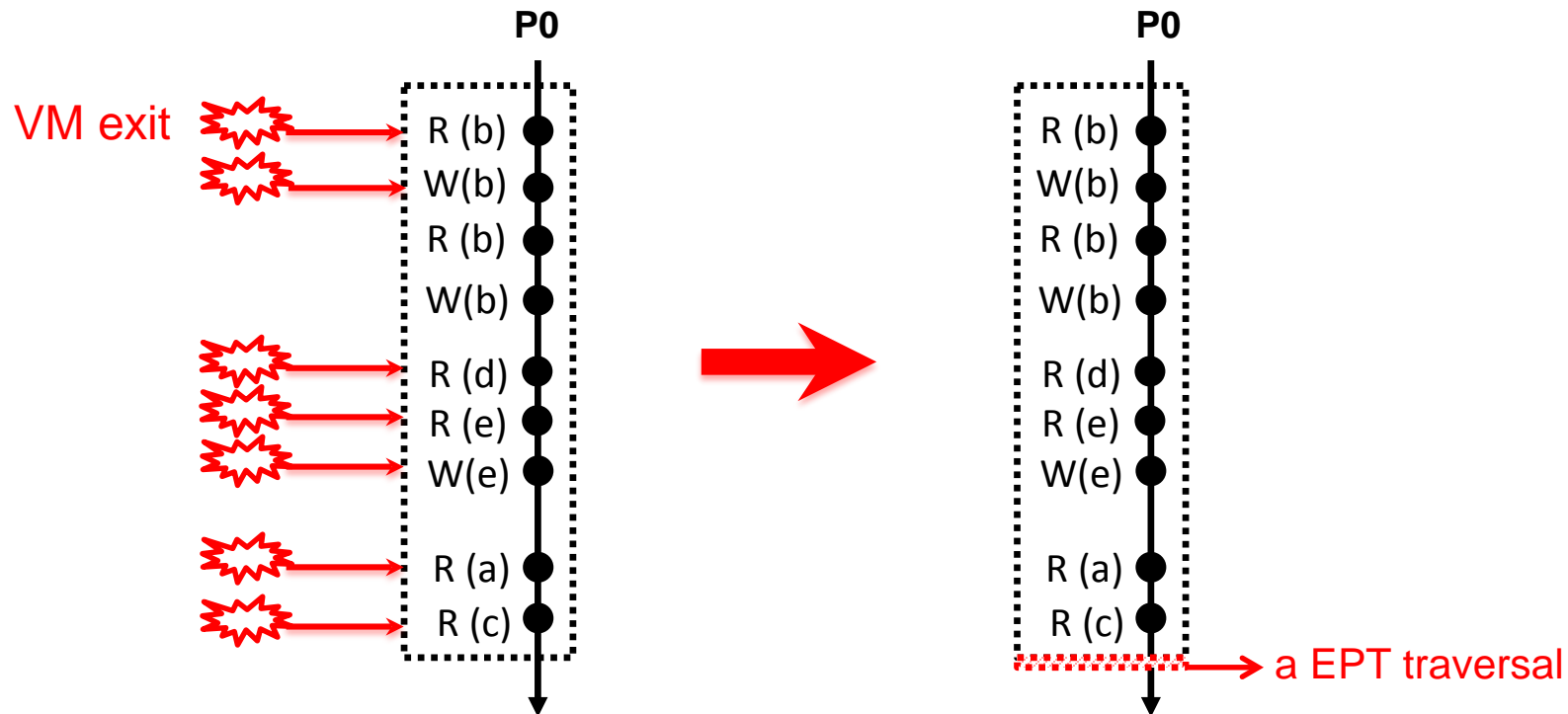
## Obtain R&W-set Efficiently via HAV Extensions

- VM-based approaches: numerous VM exits (hardware page protection)
- Accessed and Dirty Flags of EPT (Extended Page Tables)
- Our approach: a simple EPT traversal

# Record & Replay Memory Interleaving with HAV

## Obtain R&W-set Efficiently via HAV Extensions

- VM-based approaches: numerous VM exits (hardware page protection)
- Accessed and Dirty Flags of EPT (Extended Page Tables)
- Our approach: a simple EPT traversal

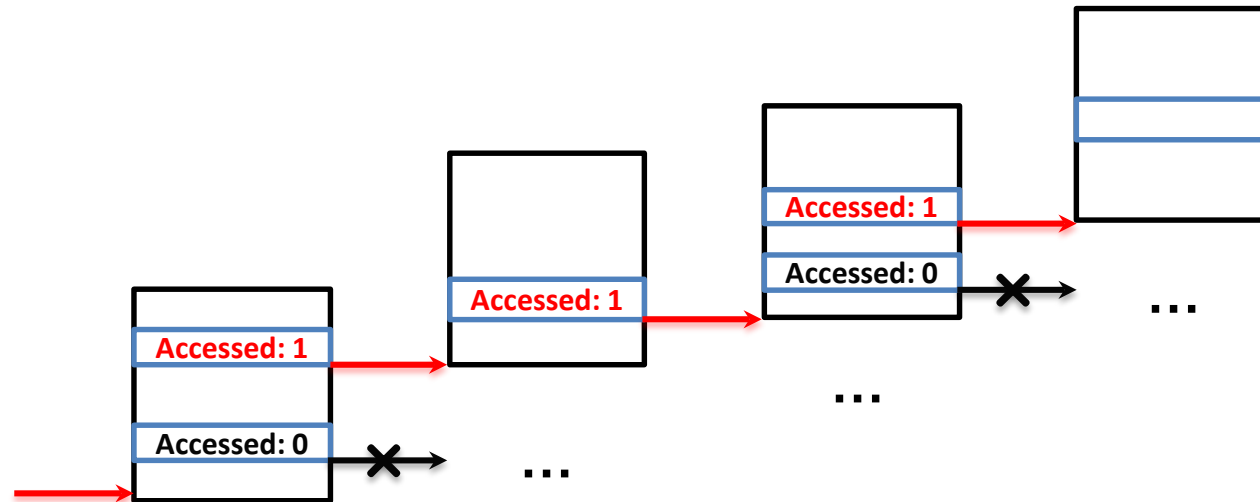




# Record & Replay Memory Interleaving with HAV

## Partial traversal of EPT

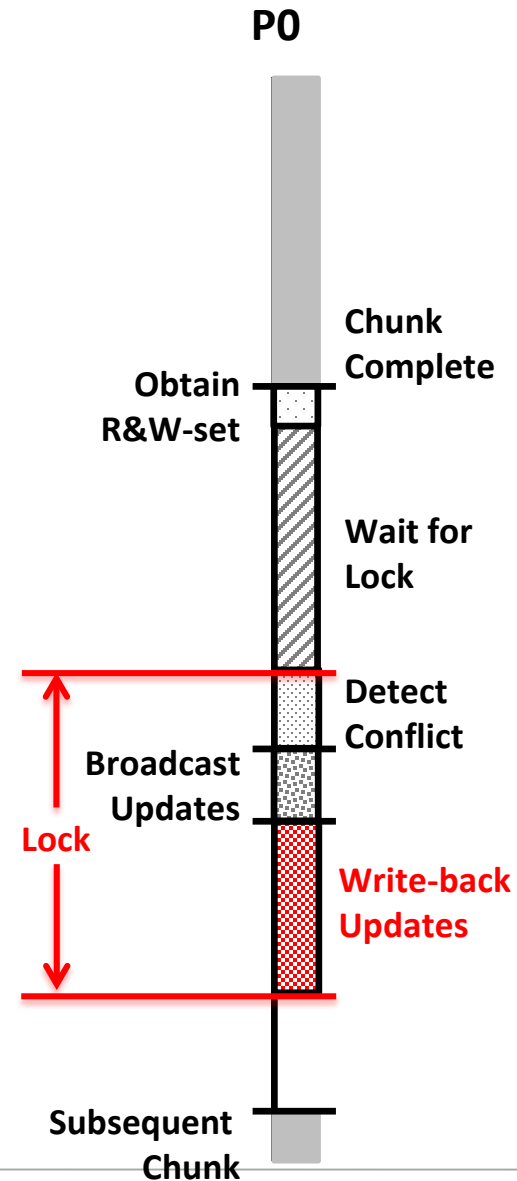
- EPT uses a hierarchical, tree-based design
- If the accessed flag of one internal entry is 0, then the accessed flags of all entries in its subtrees are definitely 0
- Locality of reference (traverse a tiny part of EPT)



# Record & Replay Memory Interleaving with HAV

## Observations

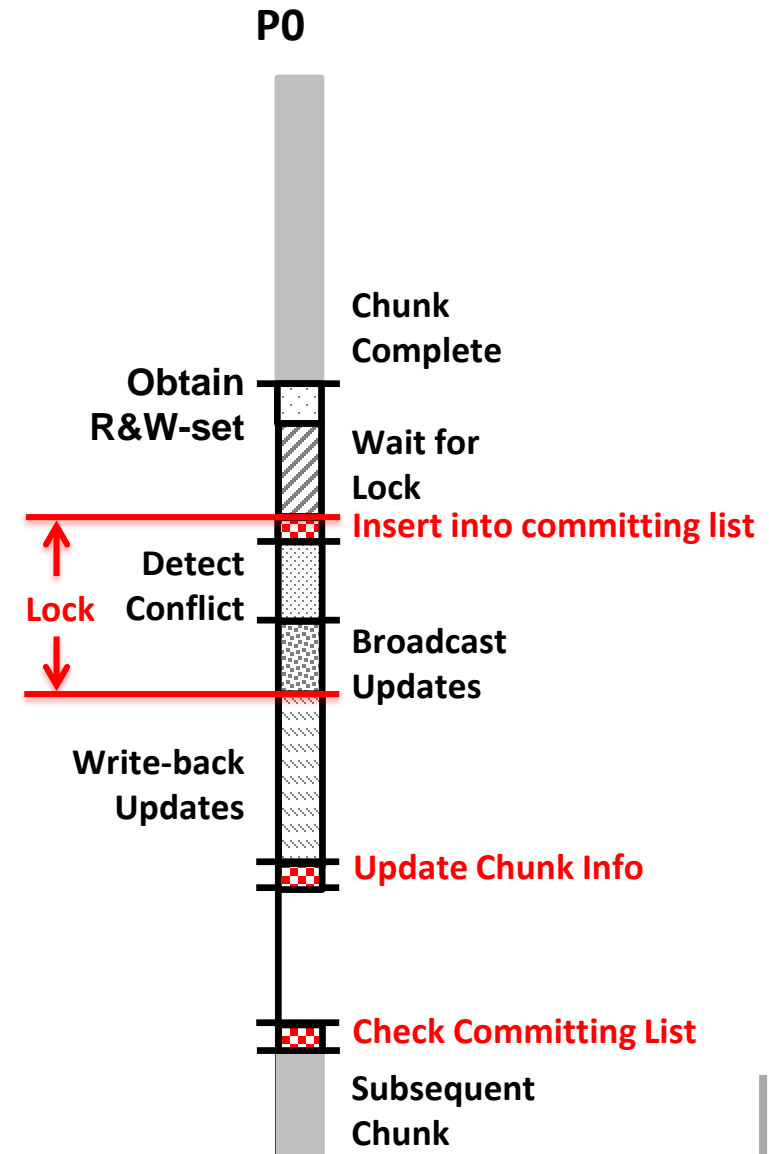
- Chunk commit is time-consuming
- Wait for lock
- Write-back operation



# Record & Replay Memory Interleaving with HAV

## Decentralized Three-Phase Commit Protocol

- Move this out of the synchronized block
- Support parallel commit while ensuring serializability
- Three phases:
  - Pre-commit phase
  - Commit phase
  - Synchronization phase

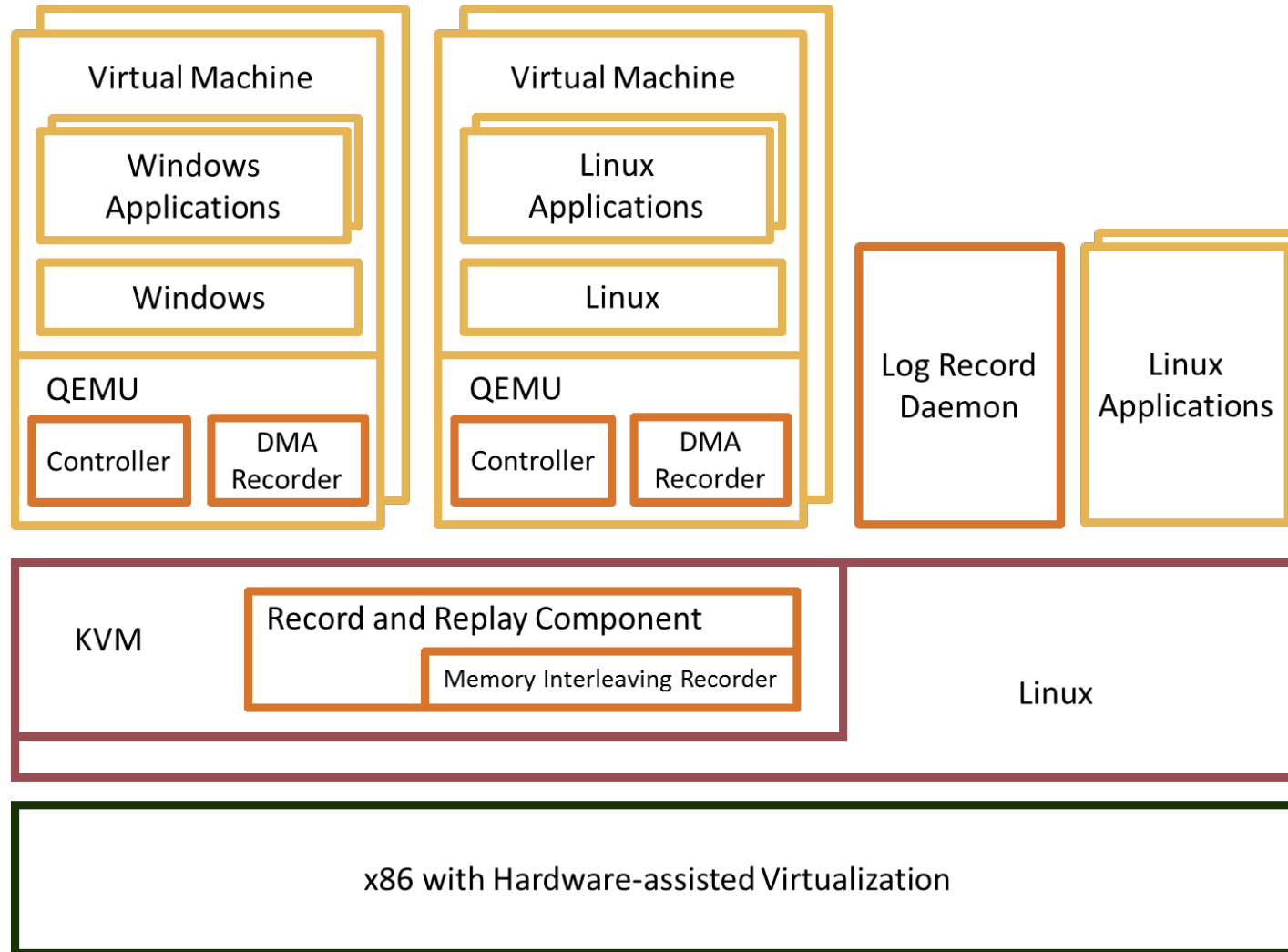


# Record & Replay Memory Interleaving with HAV

## Replay Memory Interleaving

- Guarantee all chunks will be properly re-built and executed in the original order
- Design goal: maintain the same parallelism as the recoding phase
  - 1. Truncate a chunk at the recorded timestamp
  - 2. Ensure that all preceding chunks have been committed successfully before the current chunk starts

# Samsara Overview



# Evaluation

## Experimental Setup

- 4-core Intel Core i7-4790 processor, 12GB memory, 1TB Hard Drive
- Host: Ubuntu 12.04 with Linux kernel version 3.11.0 and Qemu-1.2.2
- Guest: Ubuntu 14.04 with Linux kernel version 3.13.1

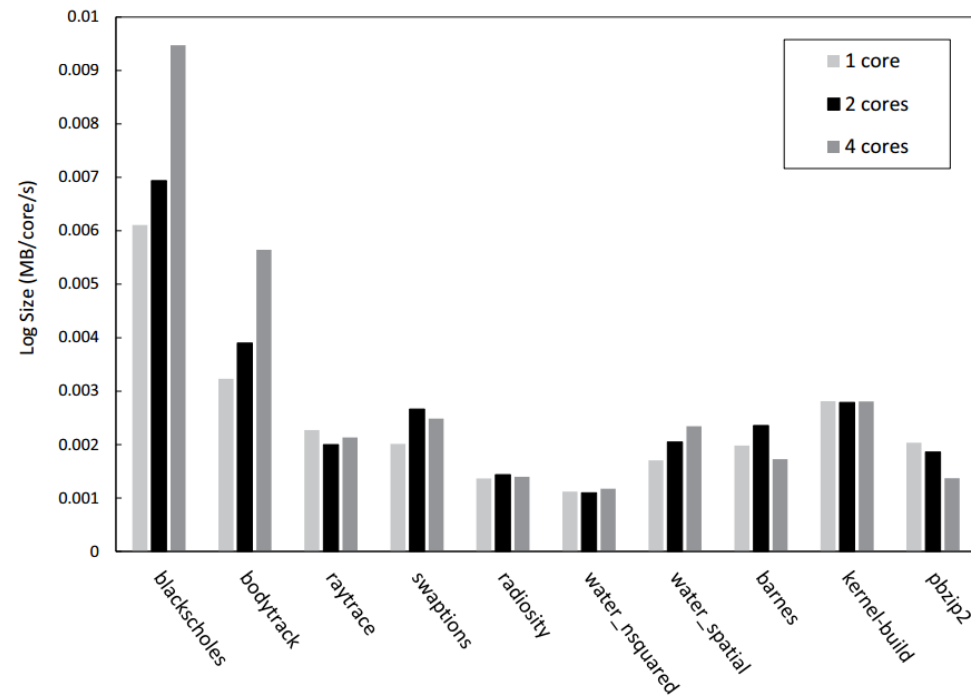
## Workloads

- Computation intensive applications
  - PARSEC
  - SPLASH-2
- I/O intensive applications
  - kernel-build
  - pbzip2

# Evaluation

## Log Size

- Samsara generates log at an average rate of **0.0027 MB/core/s** and **0.0031 MB/core/s** for recoding two and four cores
- Reduces the log file size by **98.6%** compared to the previous software-only schemes

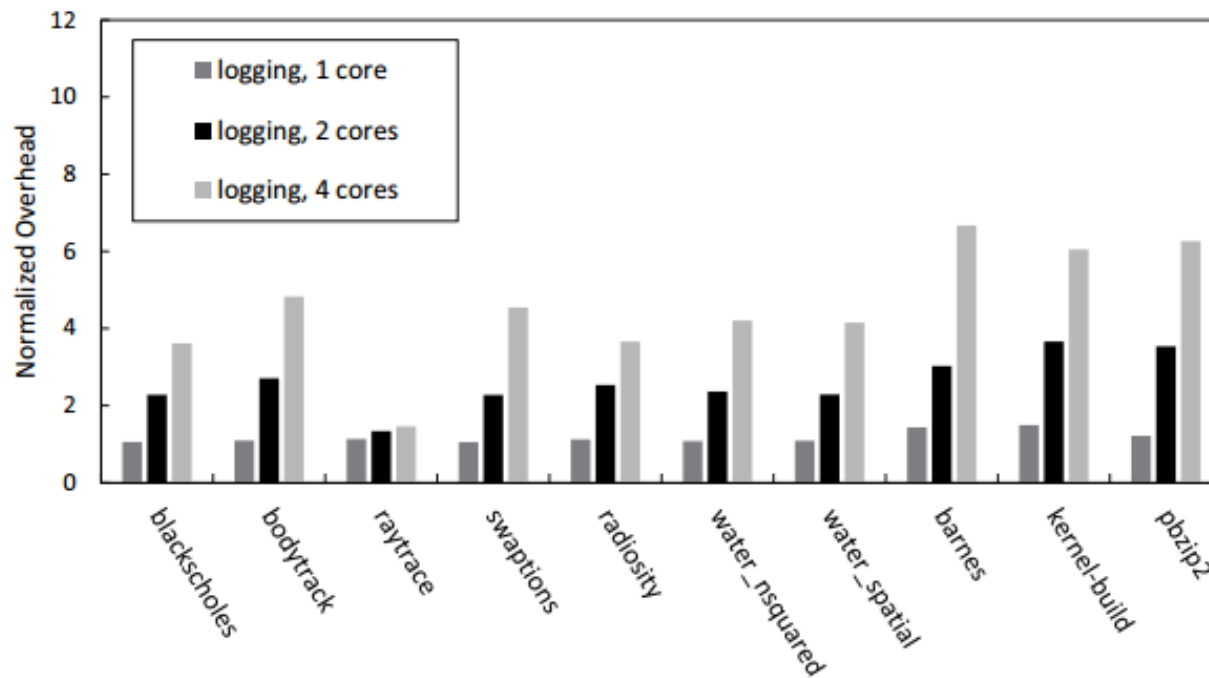


Log size produced by Samsara during recording (compressed with gzip).

# Evaluation

## Recording Overhead Compared to Native Execution

- Compare the performance to native execution
- **2.3X** and **4.1X** for recording these workloads on two and four cores
- Previous software-only approaches cause about **10X** with two cores



Recording overhead compared to the native execution.



# Conclusion

**We made the first attempt to leverage HAV extensions to achieve an efficient software-based replay system on commodity multiprocessors.**

- We abandon the inefficient CREW protocol and instead use a chunk-based strategy.
- We avoid all memory access detections, and obtain each chunk's read-set and write-set by retrieving the accessed and the dirty flags of the EPT.
- We propose a decentralized three-phase commit protocol which significantly reduces the performance overhead by allowing chunk commits in parallel while still ensuring serializability.



---

Thanks

Contact: [xiaozhen@pku.edu.cn](mailto:xiaozhen@pku.edu.cn)