ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems

Daniel Sanchez
MIT

Christos Kozyrakis
Stanford

ISCA-40
June 27, 2013

Introduction

- Current detailed simulators are slow (~200 KIPS)

- Simulation performance wall
  - More complex targets (multicore, memory hierarchy, …)
  - Hard to parallelize

- Problem: Time to simulate 1000 cores @ 2GHz for 1s at
  - 200 KIPS: **4 months**
  - 200 MIPS: **3 hours**

- Alternatives?
  - FPGAs: Fast, good progress, but still hard to use
  - Simplified/abstract models: Fast but inaccurate

ZSim Techniques

- Three techniques to make 1000-core simulation practical:
  1. Detailed DBT-accelerated core models to speed up sequential simulation
  2. Bound-weave to scale parallel simulation
  3. Lightweight user-level virtualization to bridge user-level/full-system gap

- ZSim achieves high performance and accuracy:
  - Simulates 1024-core systems at 10s-1000s of MIPS
  - 100-1000x faster than current simulators
  - Validated against real Westmere system, avg error ~10%

This Presentation is Also a Demo!

- ZSim is simulating these slides
  - OOO cores @ 2 GHz
  - 3-level cache hierarchy

ZSim performance relevant when busy
Running 2-core laptop CPU
~12x slower than 16-core server

Idle (< 0.1 cores active)	🟩
0.1 < cores active < 0.9	🟨
Busy (> 0.9 cores active)	🟥

Total cycles and instructions simulated (in billions)

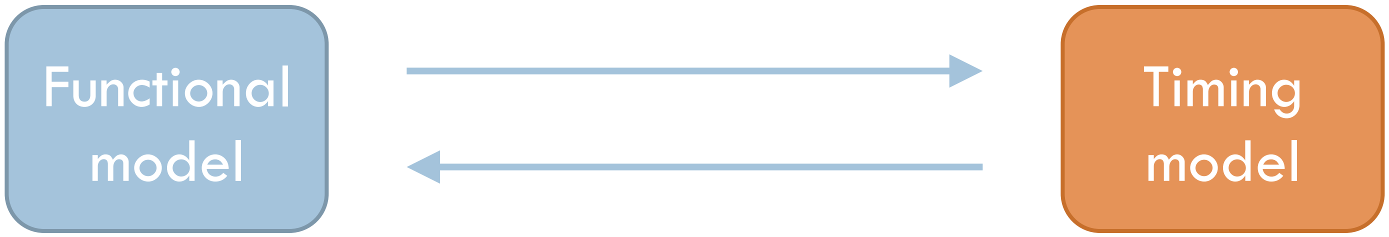
Current simulation speed and basic stats (updated every 500ms)

```
Cycles:      1.4 B    Sim Speed: 172.4 MCPS    Avg Act Cores: 1.00
Instrs:      1.3 B    Sim Speed: 169.2 MIPS    Avg Core IPC:  0.98
```

🟥

Main Design Decisions

- General execution-driven simulator:



Emulation? (e.g., gem5, MARSSx86)
Instrumentation? (e.g., Graphite, Sniper)

Dynamic Binary Translation (Pin)
✓ Functional model "for free"
✘ Base ISA = Host ISA (x86)

Cycle-driven?
Event-driven?

DBT-accelerated, instruction-driven core
+
Event-driven uncore

Outline

- Introduction
- Detailed DBT-accelerated core models
- Bound-weave parallelization
- Lightweight user-level virtualization

Accelerating Core Models

- Shift most of the work to DBT instrumentation phase

Basic block → **Instrumented basic block** + **Basic block descriptor**

```
mov   (%rbp),%rcx
add   %rax,%rbx
mov   %rdx,(%rbp)
ja   40530a
```

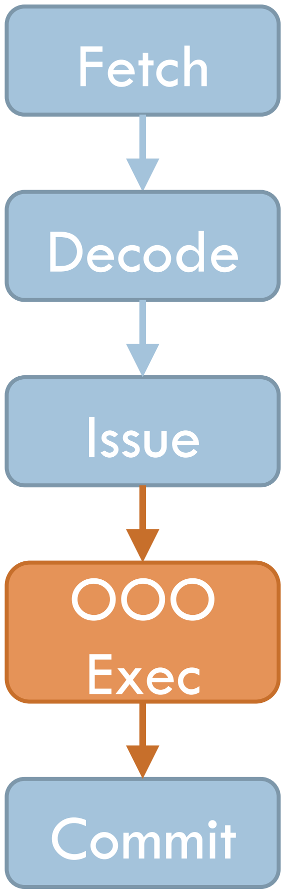
```
Load(addr = (%rbp))
mov   (%rbp),%rcx
add   %rax,%rdx
Store(addr = (%rbp))
mov   %rdx,(%rbp)
BasicBlock(BBLDescriptor)
ja   10840530a
```

Ins→µop decoding
µop dependencies,
functional units, latency
Front-end delays

- Instruction-driven models: Simulate all stages at once for each instruction/ µop
  - Accurate even with OOO if instruction window prioritizes older instructions
  - Faster, but more complex than cycle-driven
  - See paper for details

Detailed OOO Model

- OOO core modeled and validated against Westmere



Main Features

Wrong-path fetches
Branch Prediction

Front-end delays (predecoder, decoder)
Detailed instruction to μop decoding

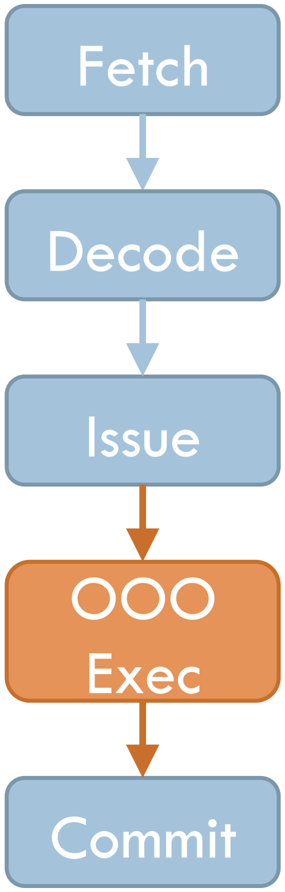
Rename/capture stalls
IW with limited size and width

Functional unit delays and contention
Detailed LSU (forwarding, fences,…)

Reorder buffer with limited size and width

Detailed OOO Model

- OOO core modeled and validated against Westmere



Fundamentally Hard to Model

Wrong-path execution

In Westmere, wrong-path instructions don’t affect recovery latency or pollute caches

Skipping OK

Not Modeled (Yet)

Rarely used instructions

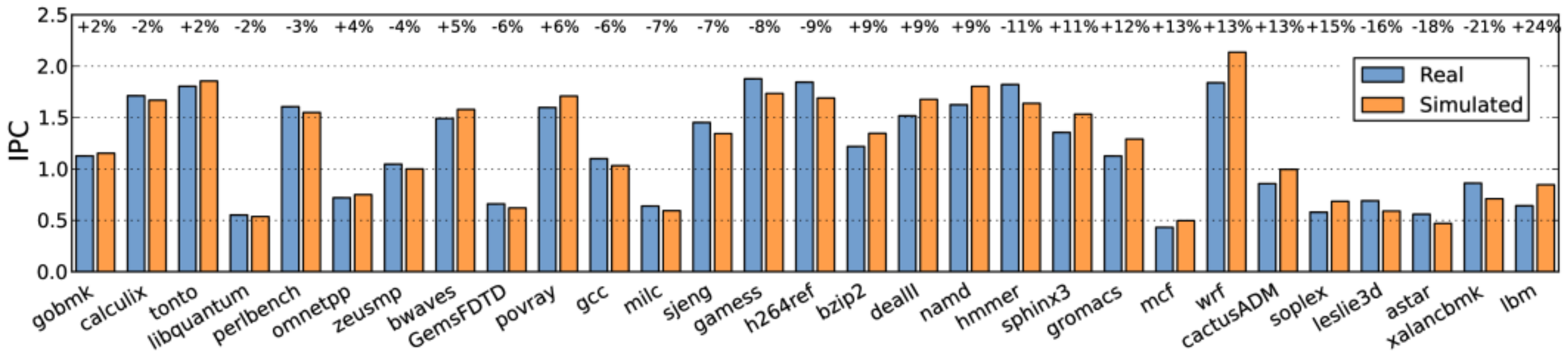
BTB

LSD

TLBs

Single-Thread Accuracy

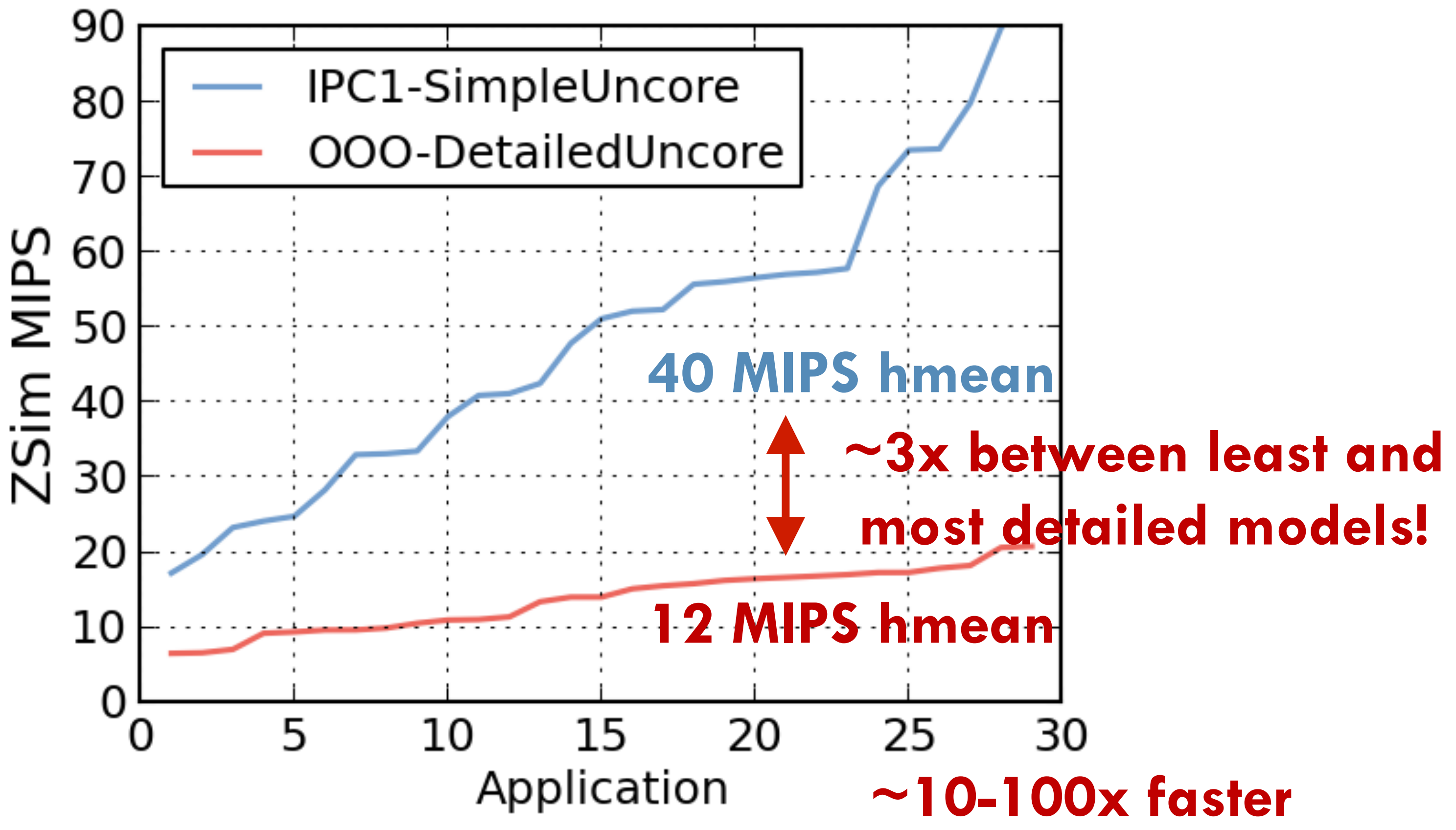
- 29 SPEC CPU2006 apps for 50 Billion instructions
- Real: Xeon L5640 (Westmere), 3x DDR3-1333, no HT
- Simulated: OOO cores @ 2.27 GHz, detailed uncore



- 9.7% average IPC error, max 24%, 18/29 within 10%

Single-Thread Performance

- Host: E5-2670 @ 2.6 GHz (single-thread simulation)
- 29 SPEC CPU2006 apps for 50 Billion instructions



~10-100x faster

Outline

- Introduction
- Detailed DBT-accelerated core models
- Bound-weave parallelization
- Lightweight user-level virtualization

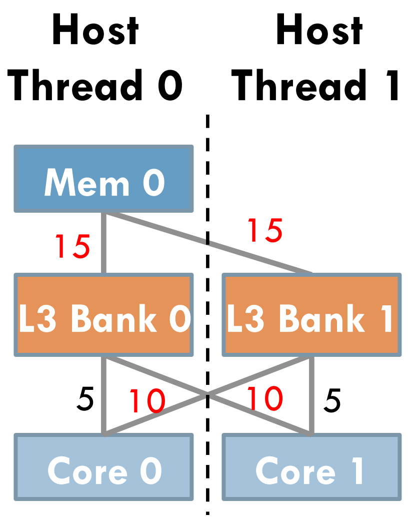
Parallelization Techniques

- Parallel Discrete Event Simulation (PDES):
  - Divide components across host threads
  - Execute events from each component maintaining illusion of full order

  ✓ Accurate

  ✘ Not scalable

Skew < 10 cycles



- Lax synchronization: Allow skews above inter-component latencies, tolerate ordering violations

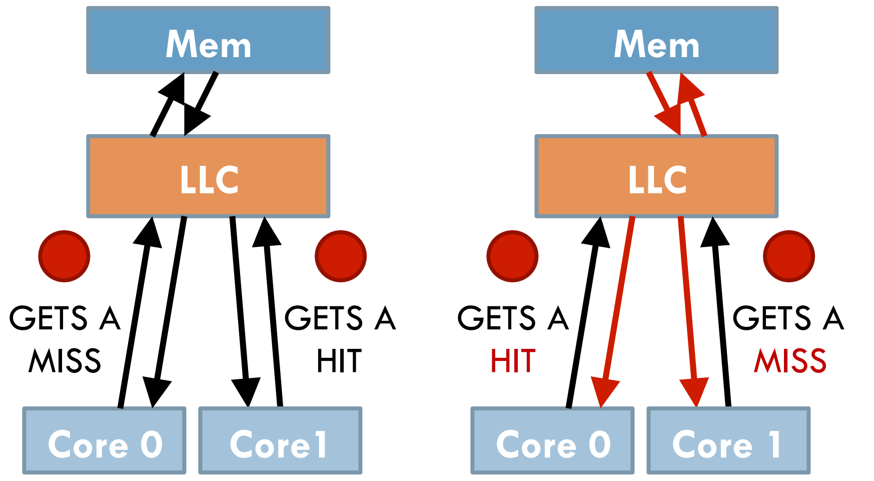
  ✓ Scalable

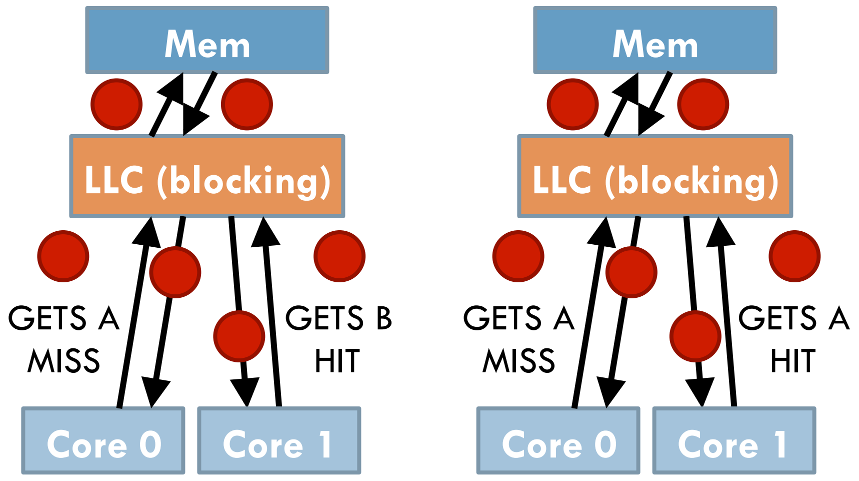
  ✘ Inaccurate

Characterizing Interference

Path-altering interference
If we simulate two accesses out of order, their paths through the memory hierarchy change

Path-preserving interference
If we simulate two accesses out of order, their timing changes but their paths do not





In small intervals (1-10K cycles), path-altering interference is extremely rare (<1 in 10K accesses)

Bound-Weave Parallelization

- Divide simulation in small intervals (e.g., 1000 cycles)
- Two parallel phases per interval: Bound and weave

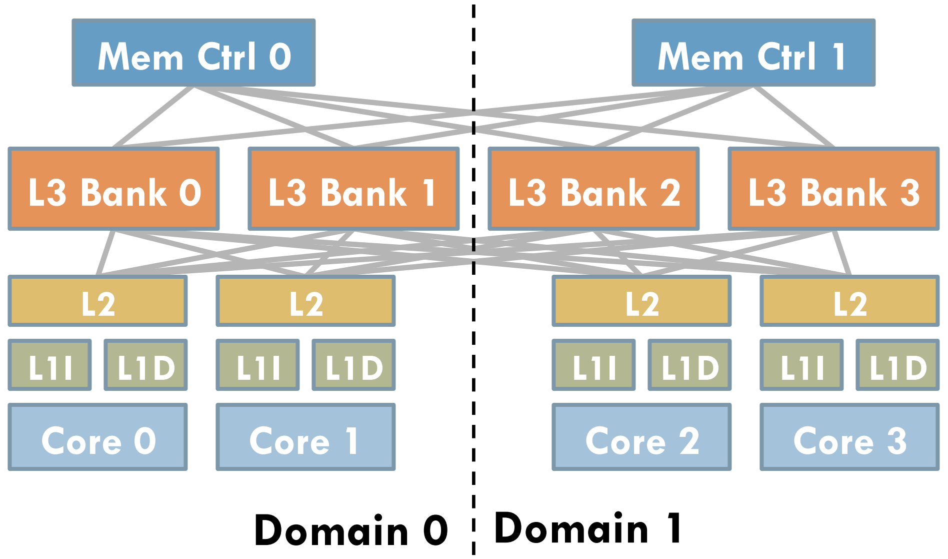
Bound phase: Find paths

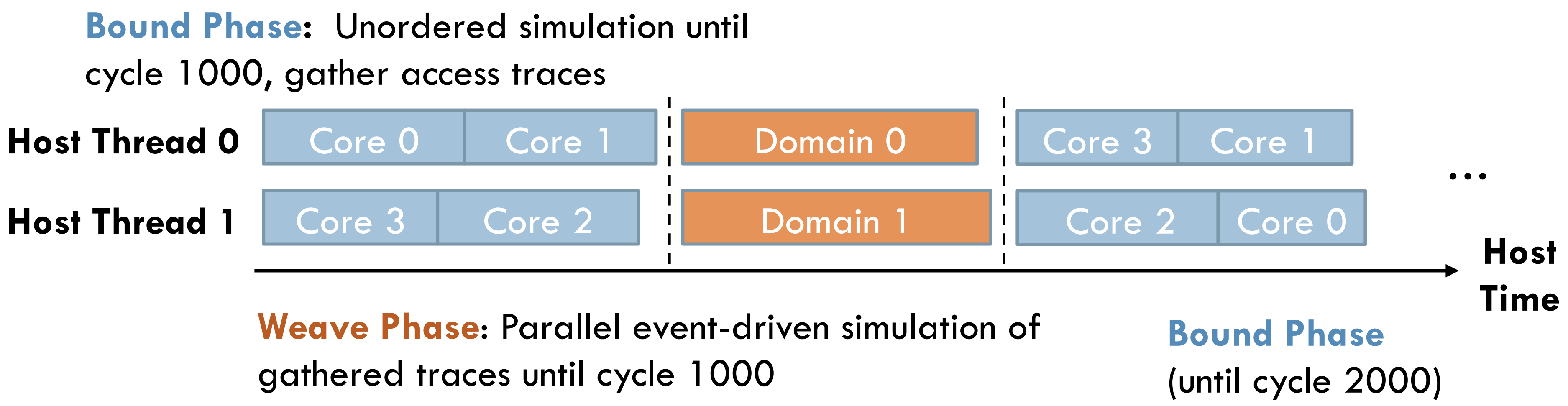
Weave phase: Find timings

Bound-Weave equivalent to PDES
for path-preserving interference

Bound-Weave Example

- 2-core host simulating 4-core system
- 1000-cycle intervals
- Divide components among 2 domains



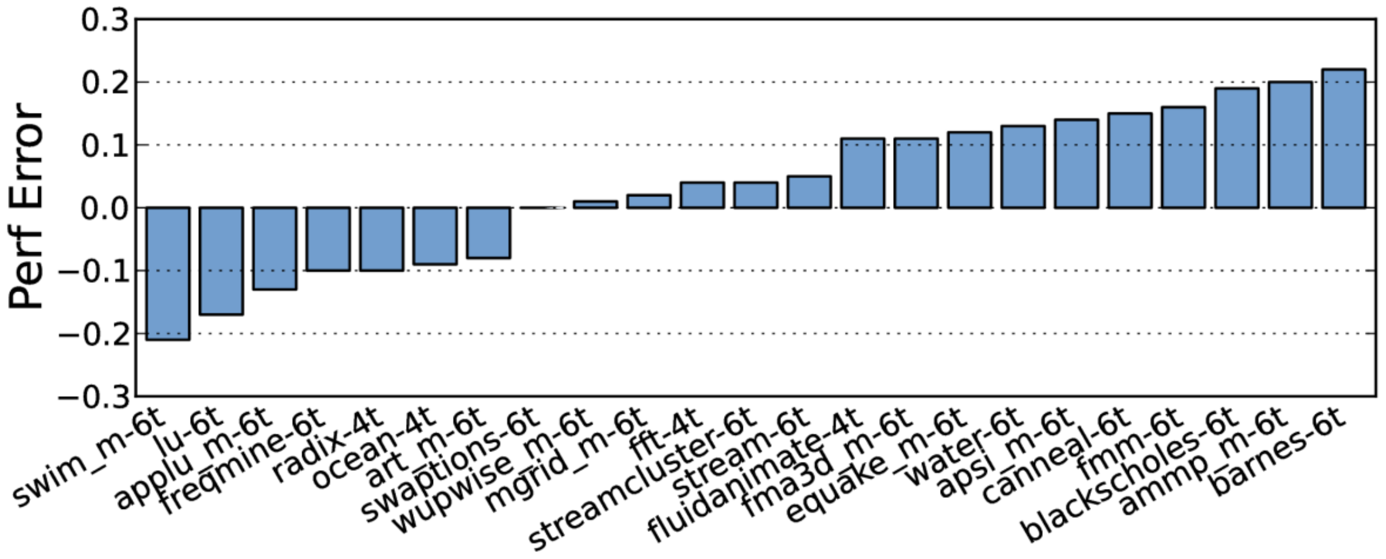

Bound Phase: Unordered simulation until cycle 1000, gather access traces
Host Thread 0
Host Thread 1
Core 0
Core 1
Core 3
Core 2
Domain 0
Domain 1
Core 3
Core 1
Core 2
Core 0
...
Host Time
Weave Phase: Parallel event-driven simulation of gathered traces until cycle 1000
Bound Phase (until cycle 2000)

Bound-Weave Take-Aways

- Minimal synchronization:
  - Bound phase: Unordered accesses (like lax)
  - Weave: Only sync on actual dependencies
- No ordering violations in weave phase
- Works with standard event-driven models
  - e.g., 110 lines to integrate with DRAMSim2
- See paper for details!

Multithreaded Accuracy

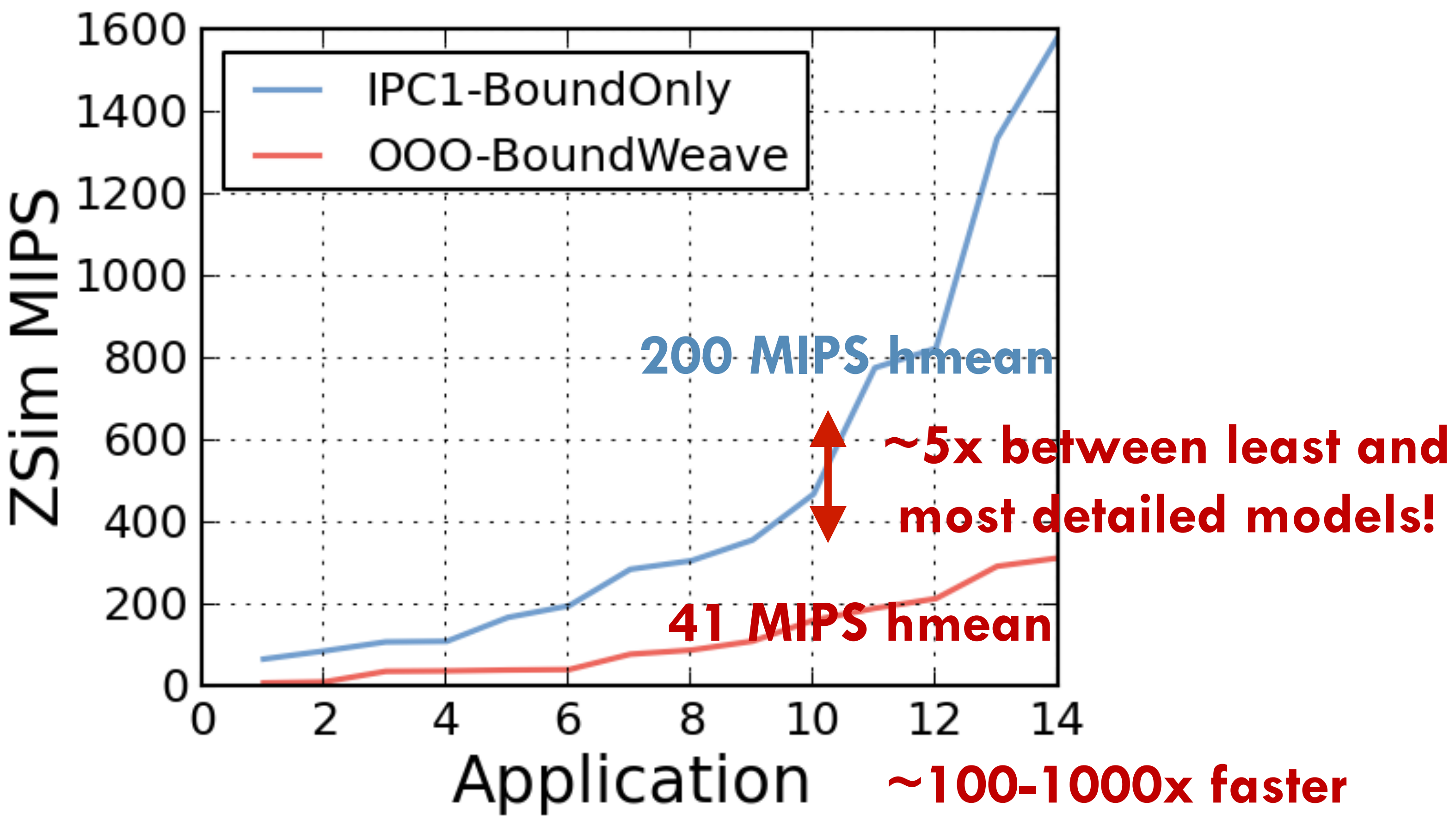
- 23 apps: PARSEC, SPLASH-2, SPEC OMP2001, STREAM



- 11.2% avg perf error (not IPC), 10/23 within 10%
  - Similar differences as single-core results
- Scalability, contention model validation → see paper

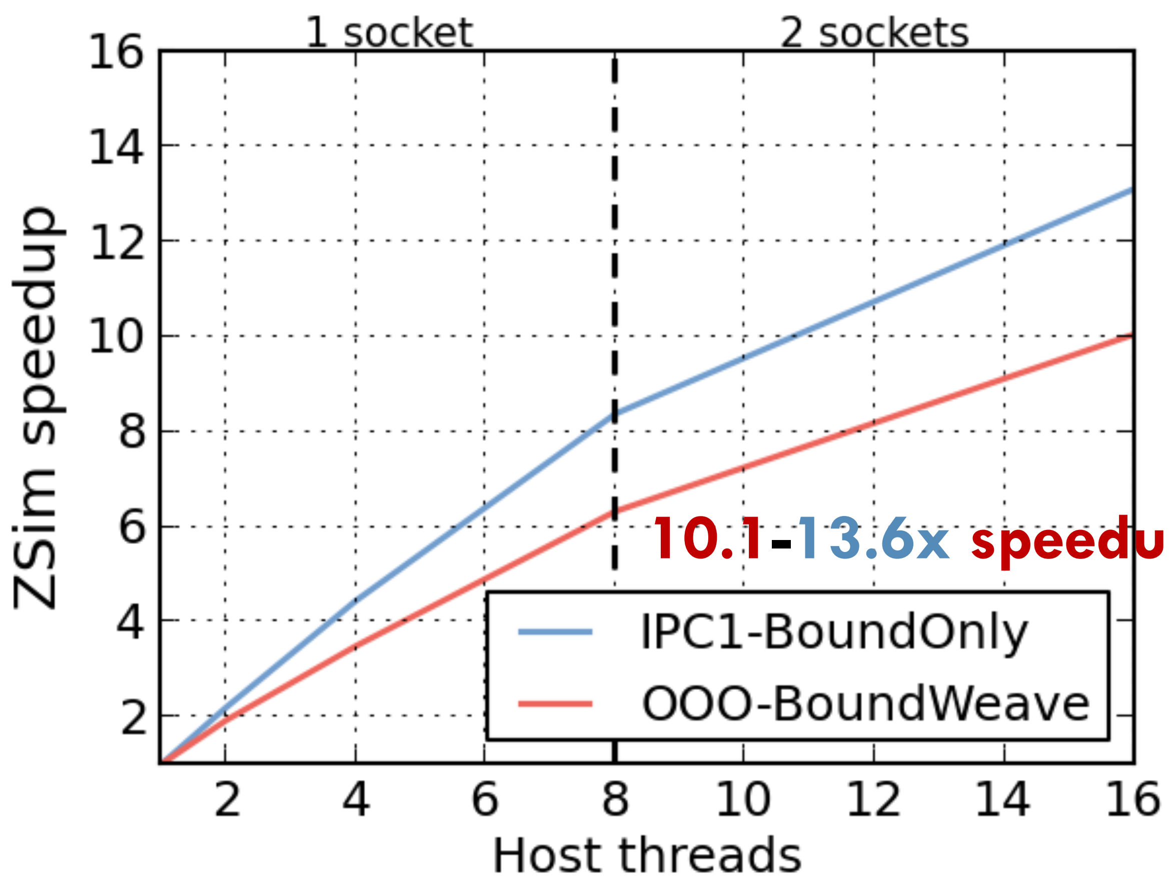
1024-Core Performance

- Host: 2-socket Sandy Bridge @ 2.6 GHz (16 cores, 32 threads)
- Results for the 14/23 parallel apps that scale



~100-1000x faster

Bound-Weave Scalability



10.1-13.6x speedup @ 16 cores

Outline

- Introduction
- Detailed DBT-accelerated core models
- Bound-weave parallelization
- Lightweight user-level virtualization

Lightweight User-Level Virtualization

- No 1Kcore OSs
- No parallel full-system DBT

$\Big\}$ ZSim has to be user-level for now

- Problem: User-level simulators limited to simple workloads
- Lightweight user-level virtualization: Bridge the gap with full-system simulation
  - Simulate accurately if time spent in OS is minimal

Lightweight User-Level Virtualization

- Multiprocess workloads
- Scheduler (threads > cores)
- Time virtualization
- System virtualization
- See paper for:
  - Simulator-OS deadlock avoidance
  - Signals
  - ISA extensions
  - Fast-forwarding

ZSim Limitations

- Not implemented yet:
  - Multithreaded cores
  - Detailed NoC models
  - Virtual memory (TLBs)

- Fundamentally hard:
  - Simulating speculation (e.g., transactional memory)
  - Fine-grained message-passing across whole chip
  - Kernel-intensive applications

Conclusions

- Three techniques to make 1Kcore simulation practical
  - DBT-accelerated models: 10-100x faster core models
  - Bound-weave parallelization: ~10-15x speedup from parallelization with minimal accuracy loss
  - Lightweight user-level virtualization: Simulate complex workloads without full-system support

- ZSim achieves high performance and accuracy:
  - Simulates 1024-core systems at 10s-1000s of MIPS
  - Validated against real Westmere system, avg error ~10%

- Source code available soon at zsim.csail.mit.edu