Relaxed Consistency Deterministic Computer

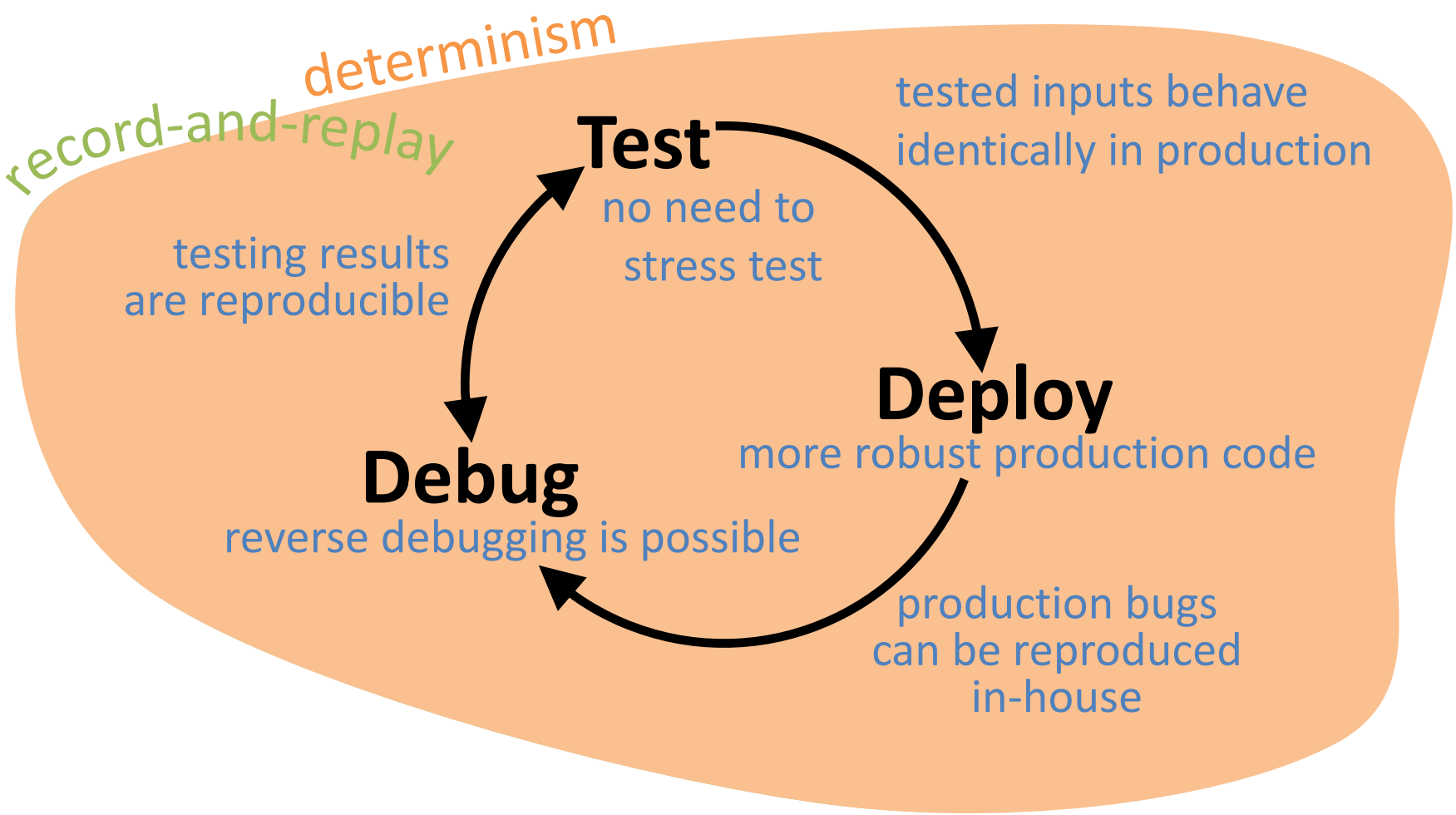
"deterministic deeds, done dirt cheap"

Joseph Devietti, Jacob Nelson, Tom Bergan
Luis Ceze, Dan Grossman

UNIVERSITY of WASHINGTON

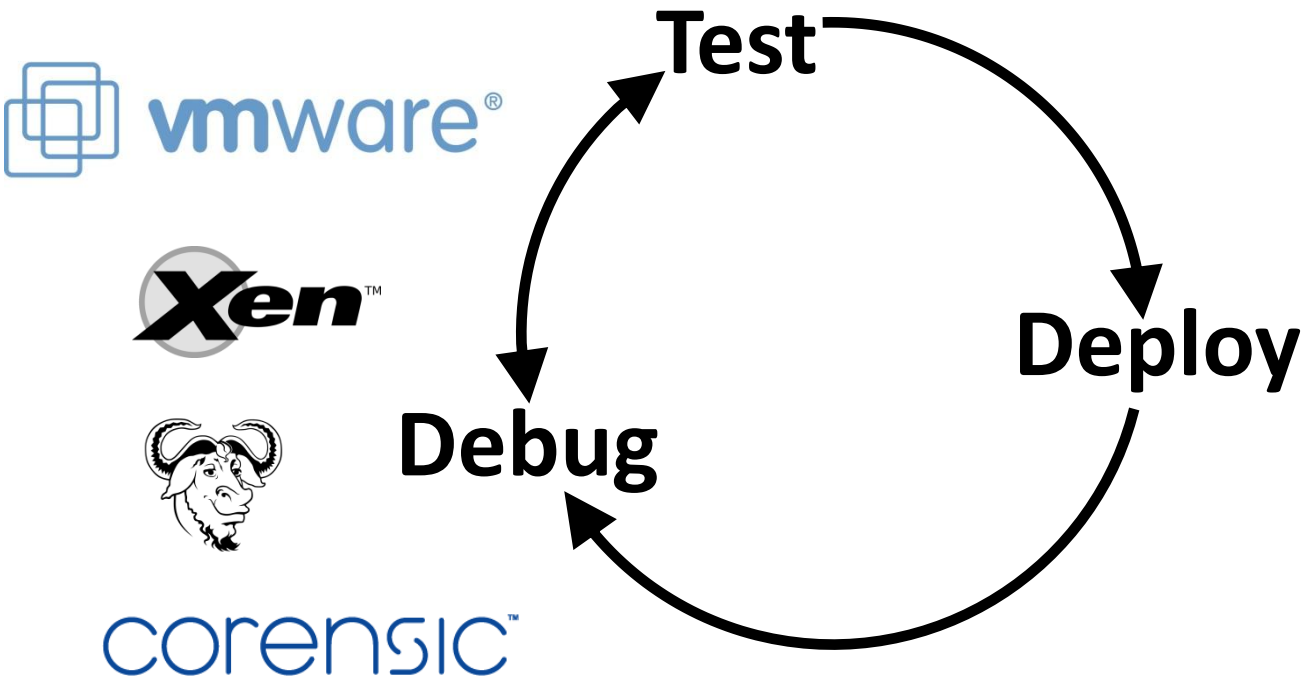
sampa

determinism improves the software development cycle



determinism

improves the software development cycle



History of Deterministic Execution

Deterministic Execution
for **Arbitrary** Programs

DMP [ASPLOS '09]
CoreDet [ASPLOS '10]
dOS [OSDI '10]
Determinator [OSDI '10]
Calvin [HPCA '11]
RC⚡DC [ASPLOS '11]

Deterministic Execution
for **Restricted** Programs

Kendo [ASPLOS '09]
Grace [OOPSLA '09]

History of Deterministic Execution

DMP [ASPLOS ‘09]

seq. consistency

CoreDet [ASPLOS ‘10]

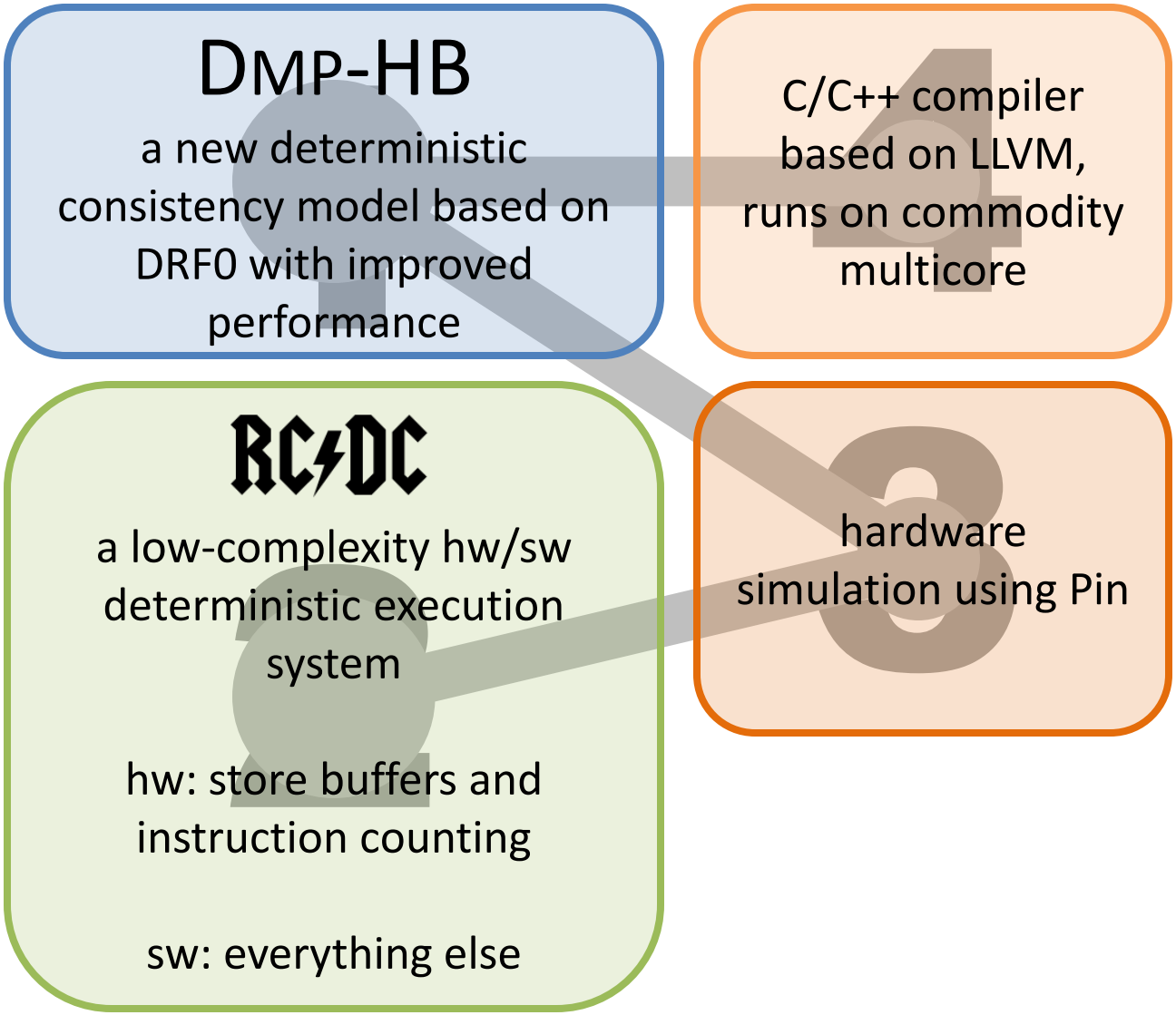
total store order

RC/DC [ASPLOS ‘11]

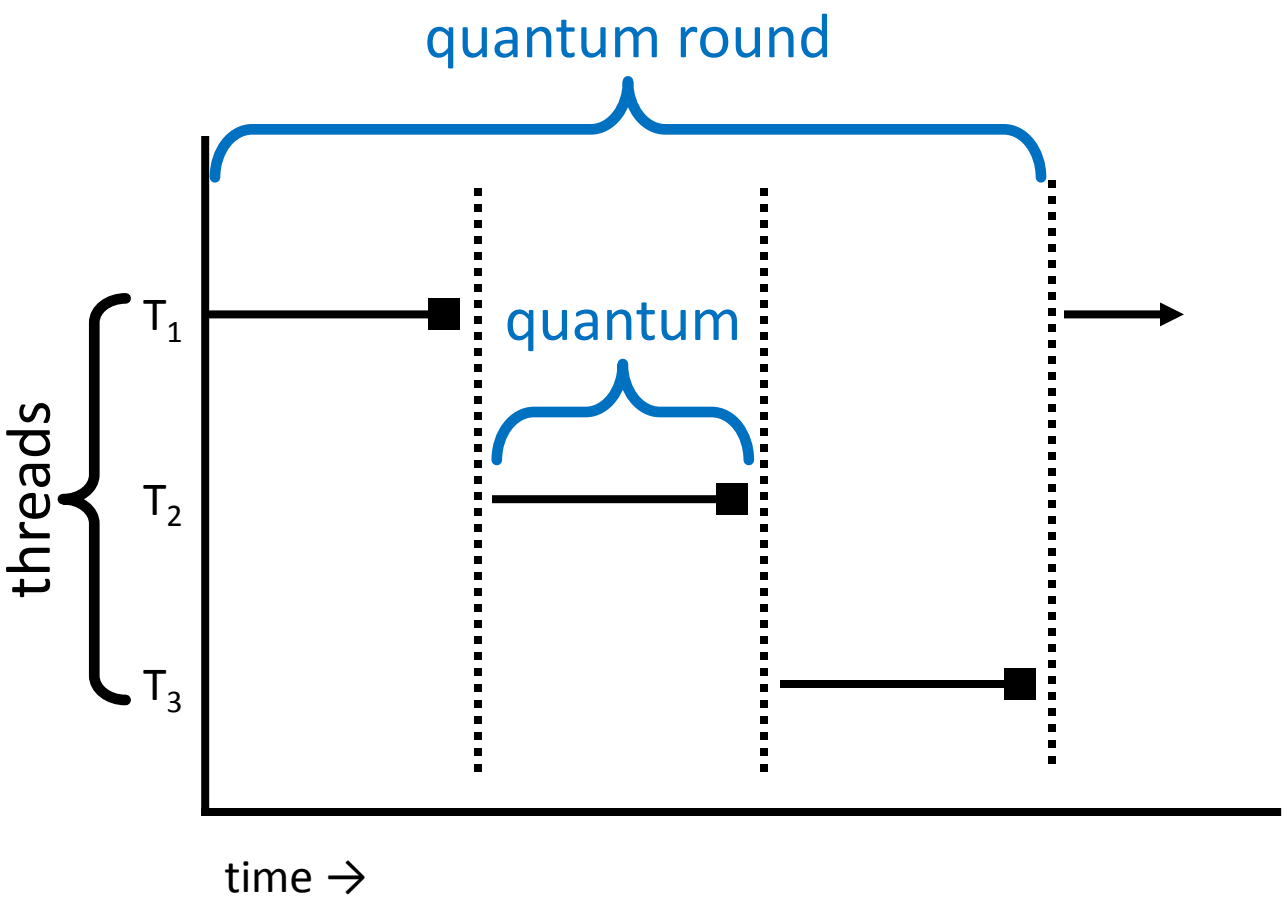
DRF0 [ISCA ‘90]

"Piled Higher and Deeper" by Jorge Cham
www.phdcomics.com

Contributions



starting simple: serialization

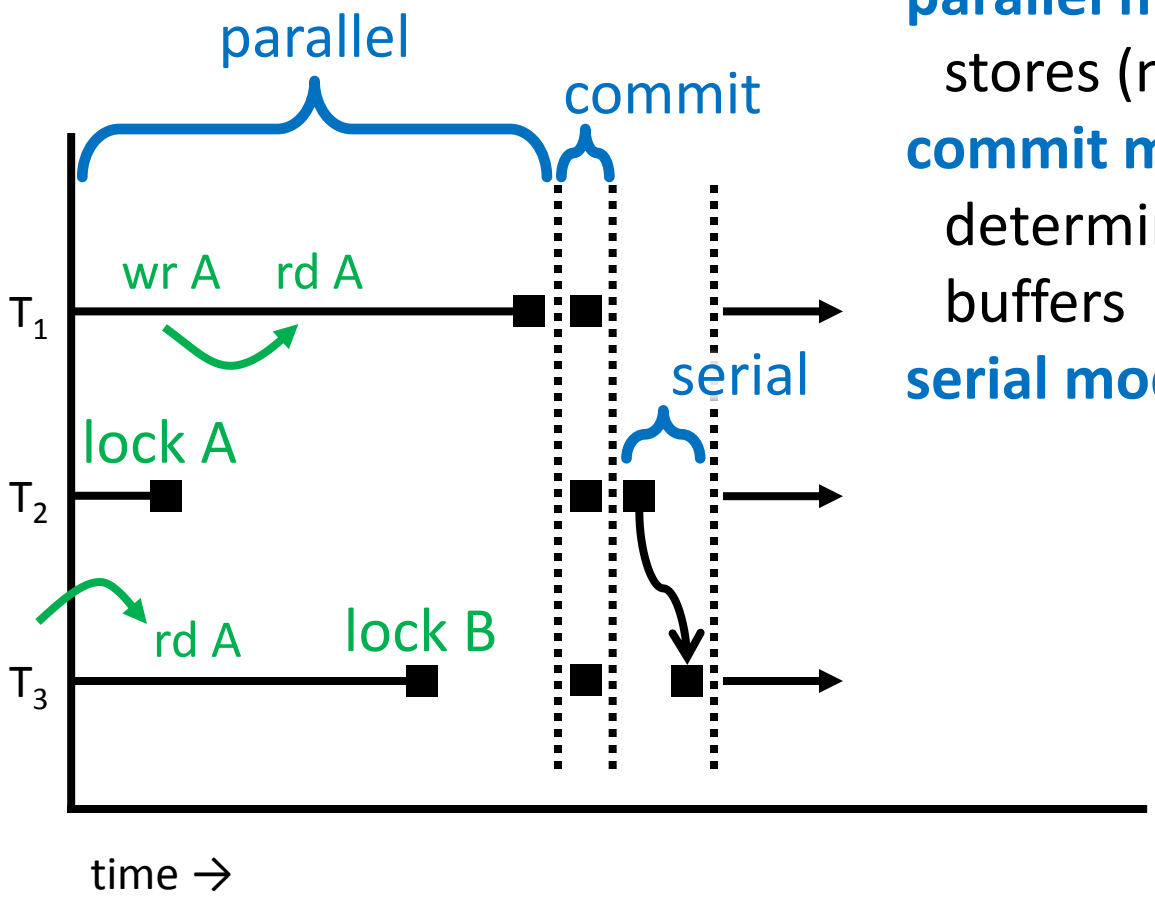


deterministic
quantum size
+
deterministic
scheduling

---

determinism

recovering parallelism with DMP-TSO

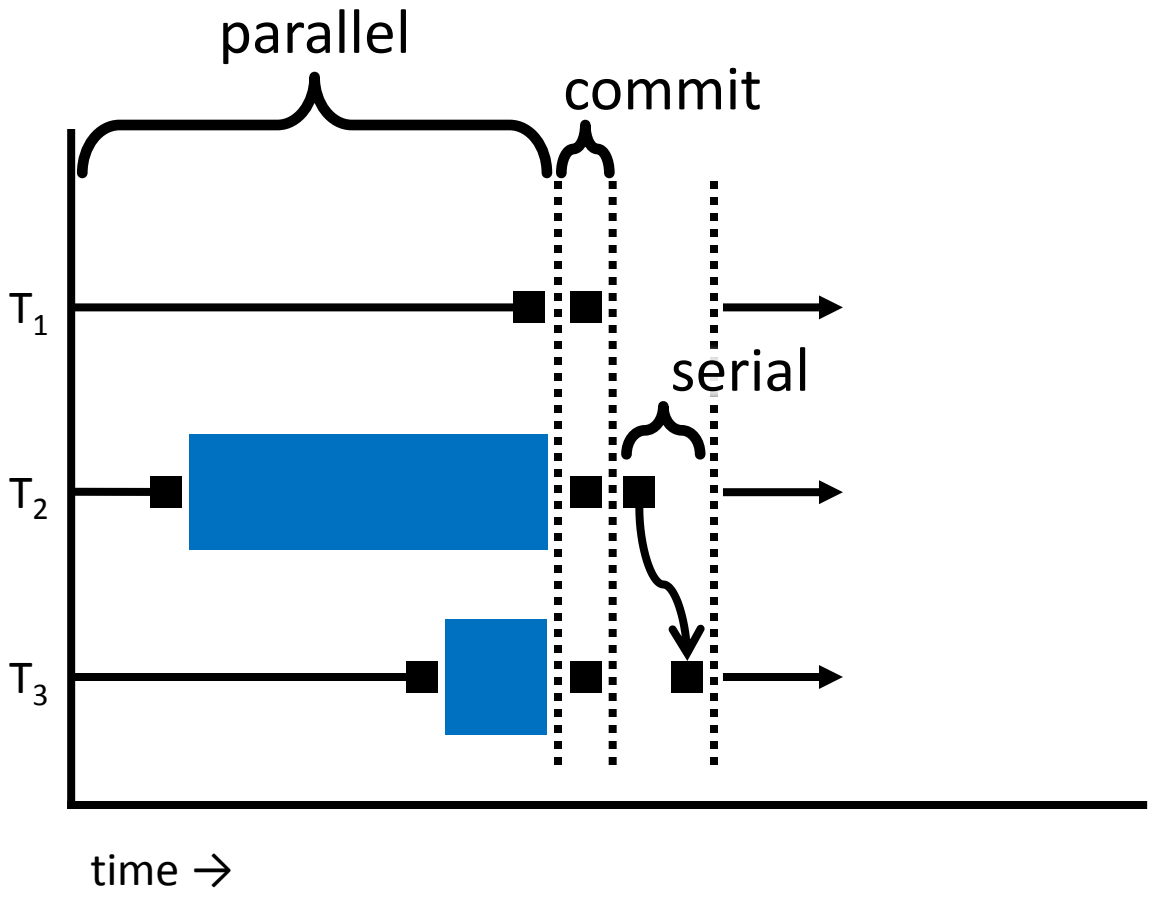


parallel mode: buffer all stores (no communication)

commit mode: deterministically publish buffers

serial mode: for atomic ops

Why is DMP-TSO slow?

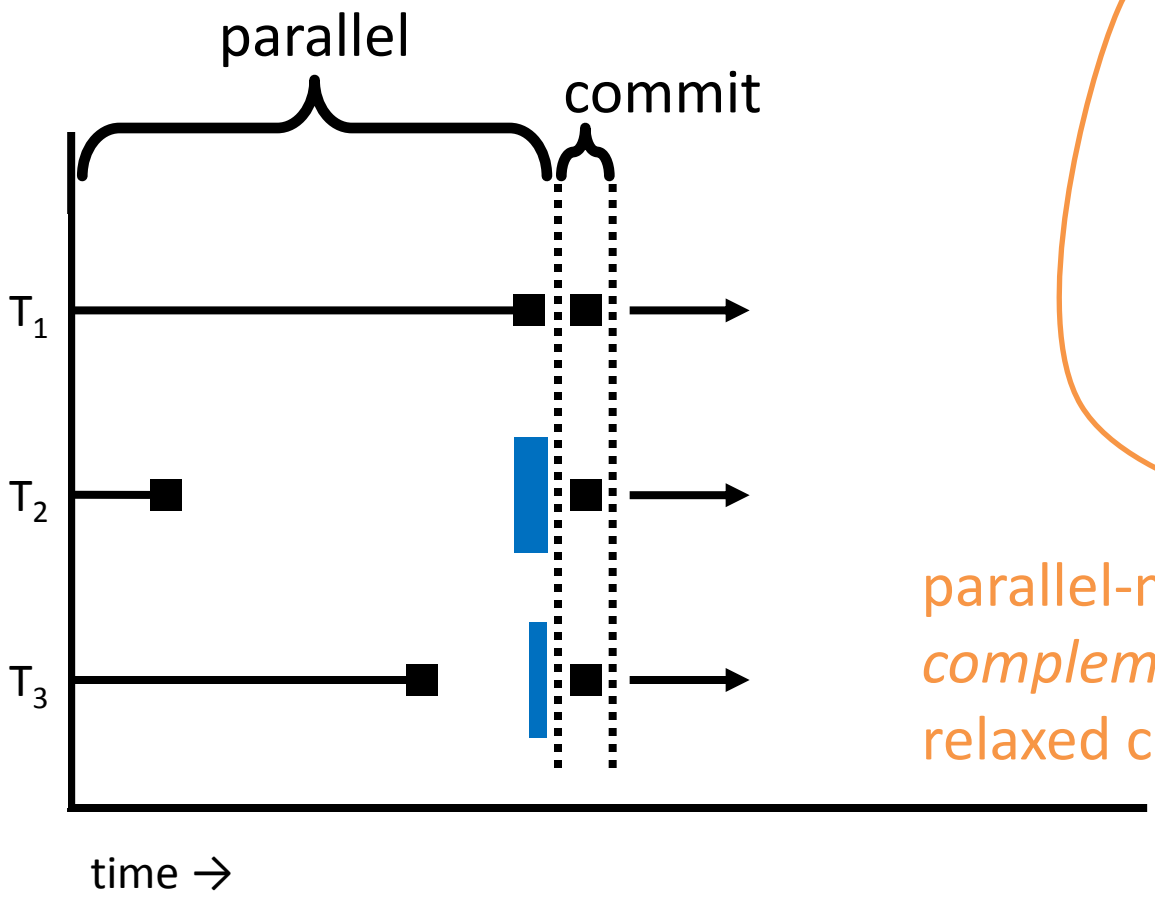


Kendo [ASPLOS '09]

~~serialization~~

imbalance

Why is DMP-TSO slow?



Kendo [ASPLOS '09]

~~serialization~~

~~imbalance~~

DMP-HB

parallel-mode synchronization

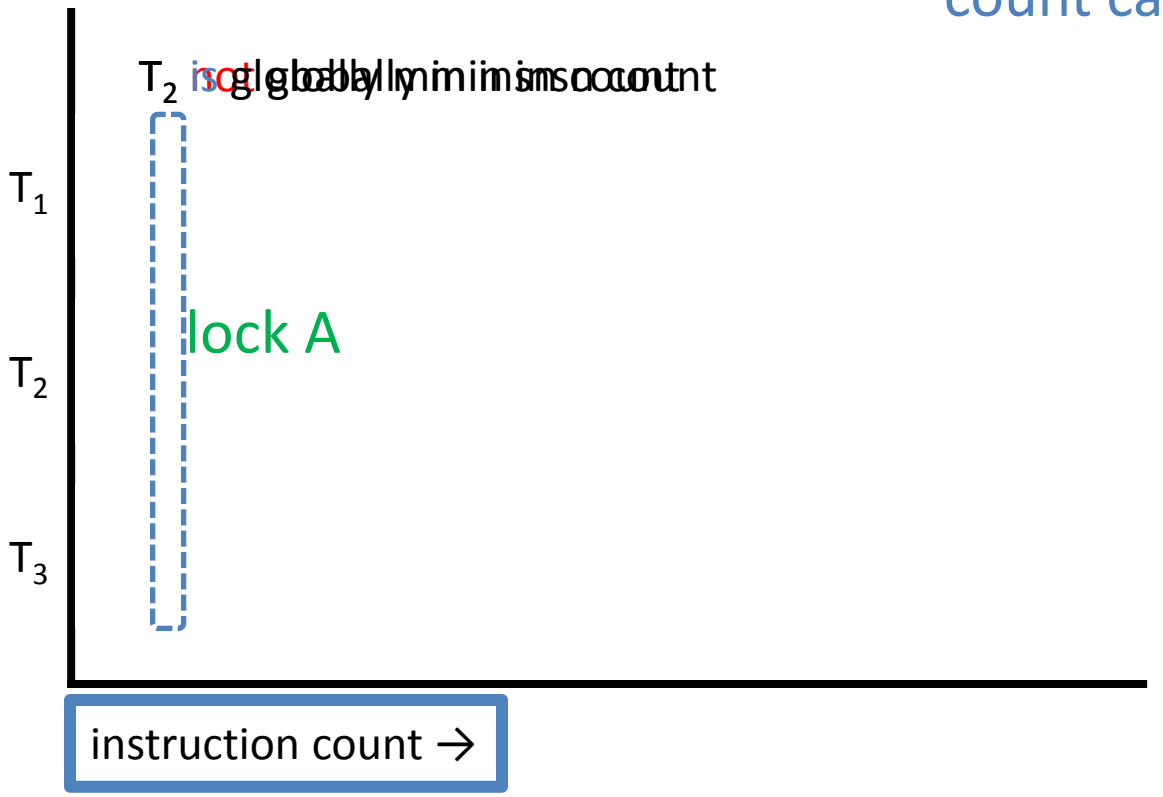
complements

relaxed consistency

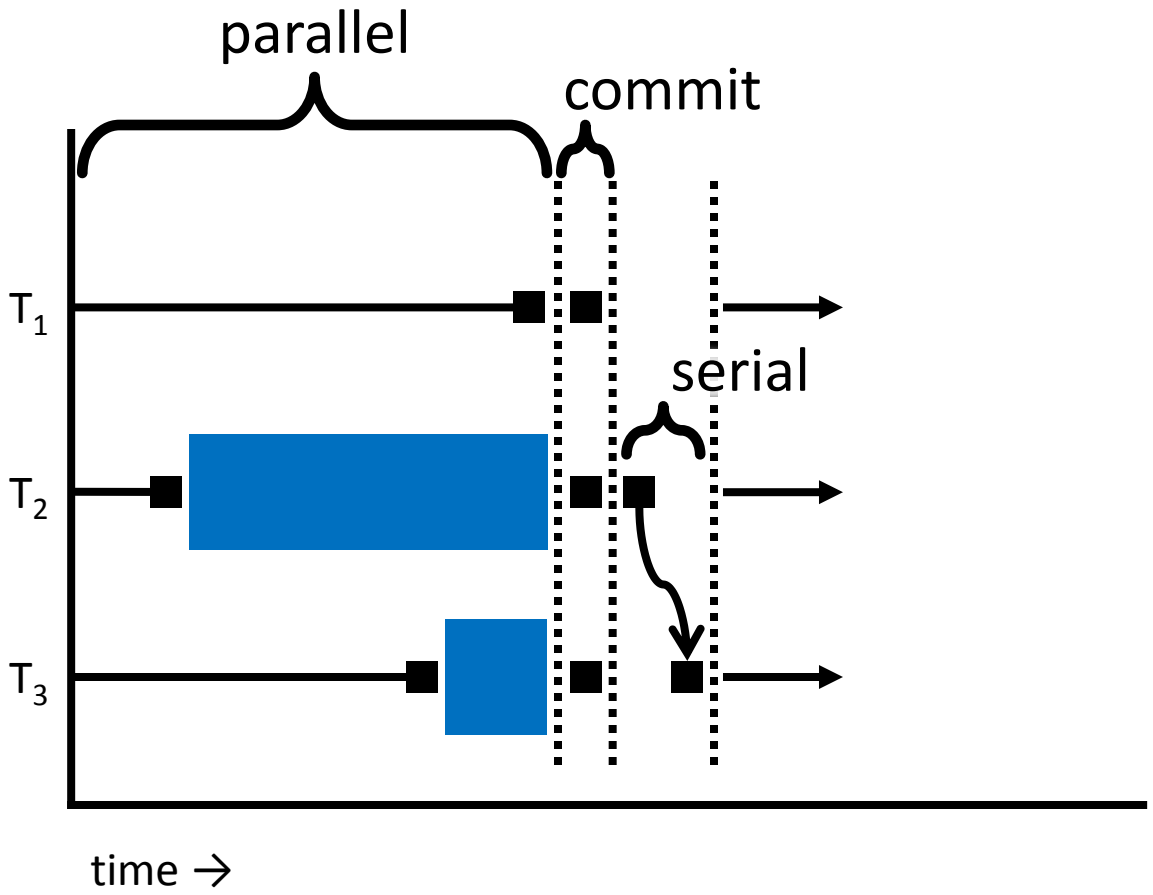
synchronization in parallel mode with Kendo

[Olszewski et al., ASPLOS '09]

thread with globally min insn count can do atomic op



Why is DMP-TSO slow?

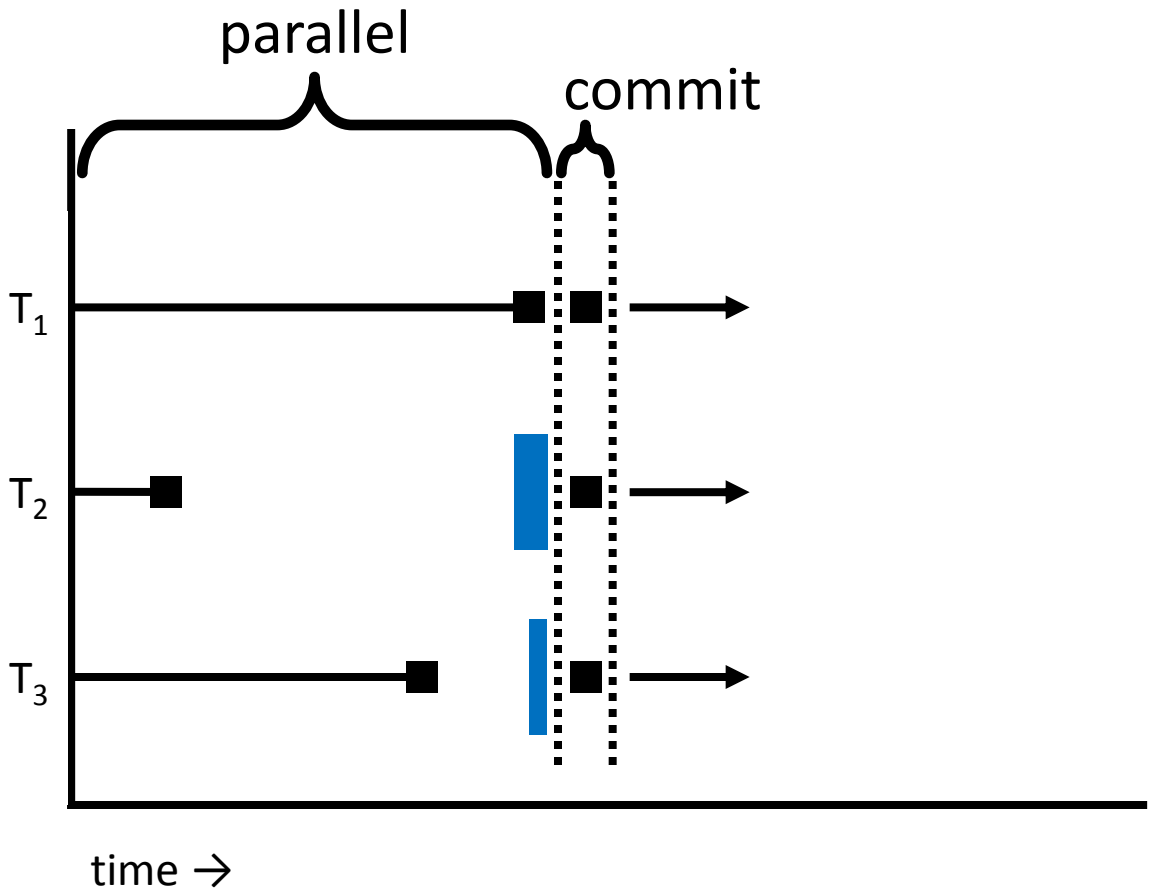


Kendo [ASPLOS '09]

~~serialization~~

imbalance

Why is DMP-TSO slow?



Kendo [ASPLOS '09]

~~serialization~~

~~imbalance~~

DMP-HB

DRF0: happens-before consistency

[Adve and Hill, ISCA '90]

- happens-before edges defined by synchronization operations
- **remote updates visible via cross-thread happens-before edges**
- SC for DRF programs
- upholds C++/Java memory models
- programmer-visible model doesn't change

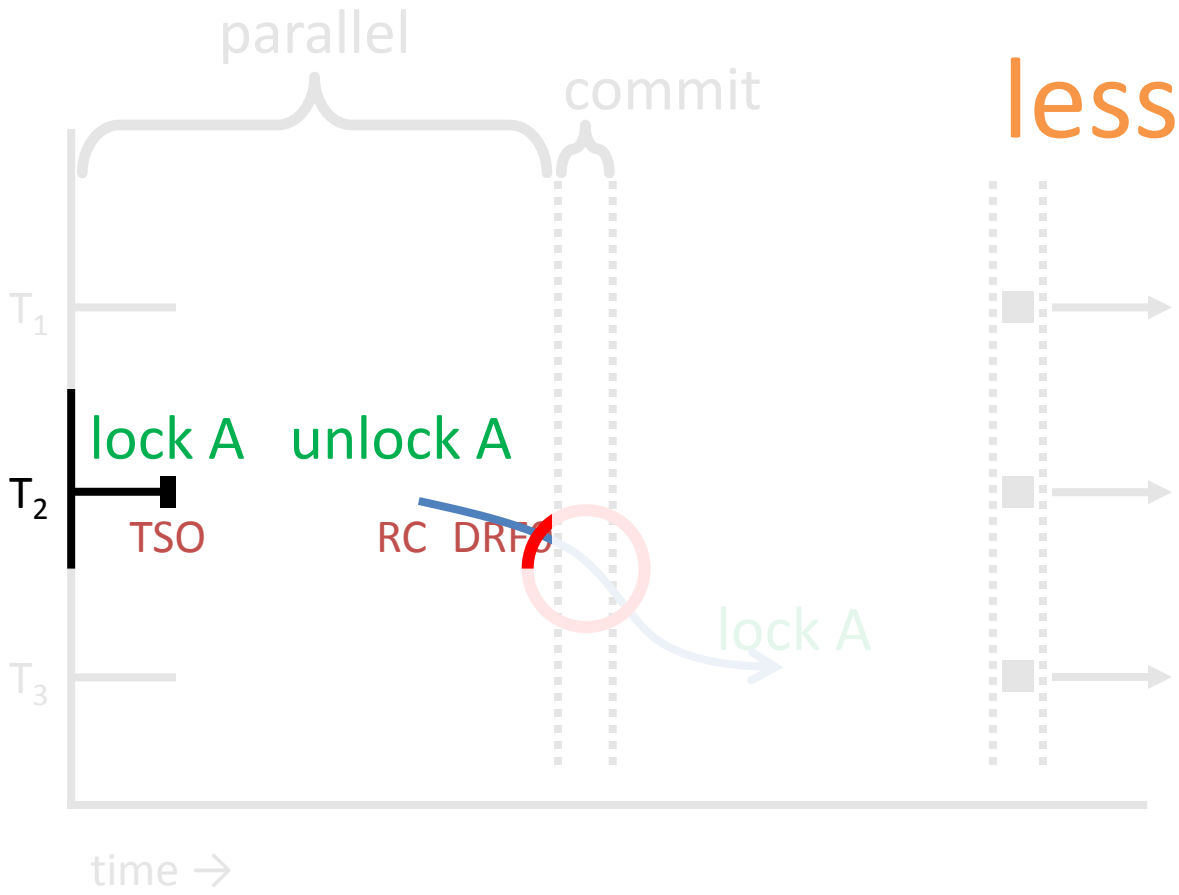
sync in parallel mode (Kendo)

relaxed consistency (DRF0)

deterministic scheduling (DMP)

DMP-HB

DMP-HB : happens-before determinism



no serial mode
less imbalance

explicit fences
rarely necessary

explicit fence iff
inter-thread HB
edge doesn't
cross commit

Outline

1

DMP-HB

a new deterministic consistency model with improved performance

4

C/C++ compiler based on LLVM, runs on commodity multicore

2

RC/DC

a low-complexity hw/sw deterministic execution system

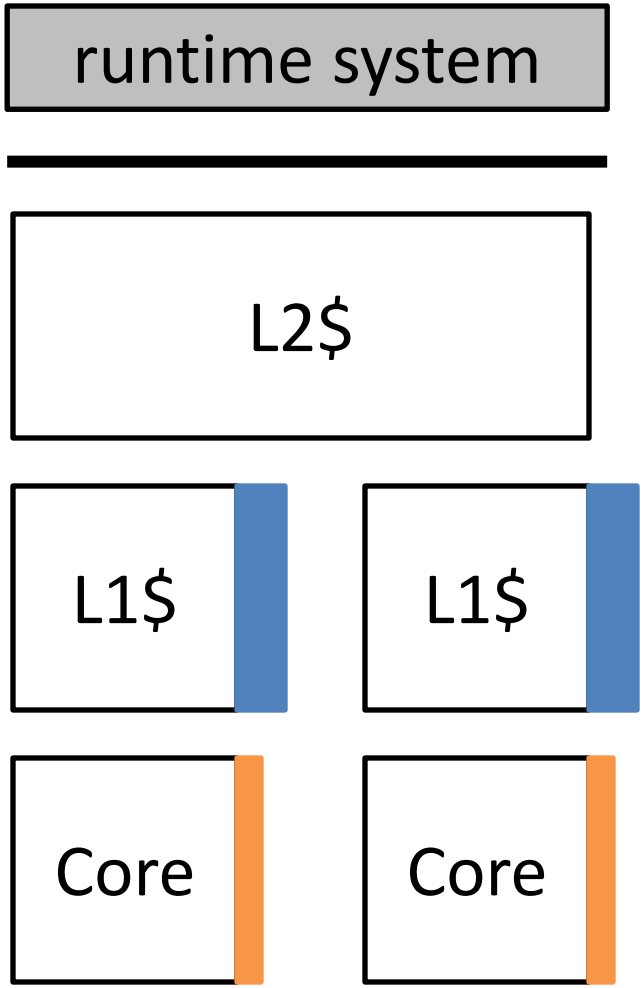
hw: store buffers and instruction counting

sw: everything else

3

hardware simulation using Pin

RC/DC Architecture



Store Buffers in Private $

application/OS can choose nondeterminism

- StoreToSB
- CommitSB

align context switches with quantum boundaries

- SaveSB
- RestoreSB

Precise Insn Counting

- StartInsnCount
- StopInsnCount
- ReadInsnCount

Traps

- SBFull
- QuantumReached

Outline

1

DMP-HB

a new deterministic consistency model with improved performance

2

RC/DC

a low-complexity hw/sw deterministic execution system

hw: store buffers and instruction counting

sw: everything else

3

hardware simulation using Pin

4

C/C++ compiler based on LLVM, runs on commodity multicore

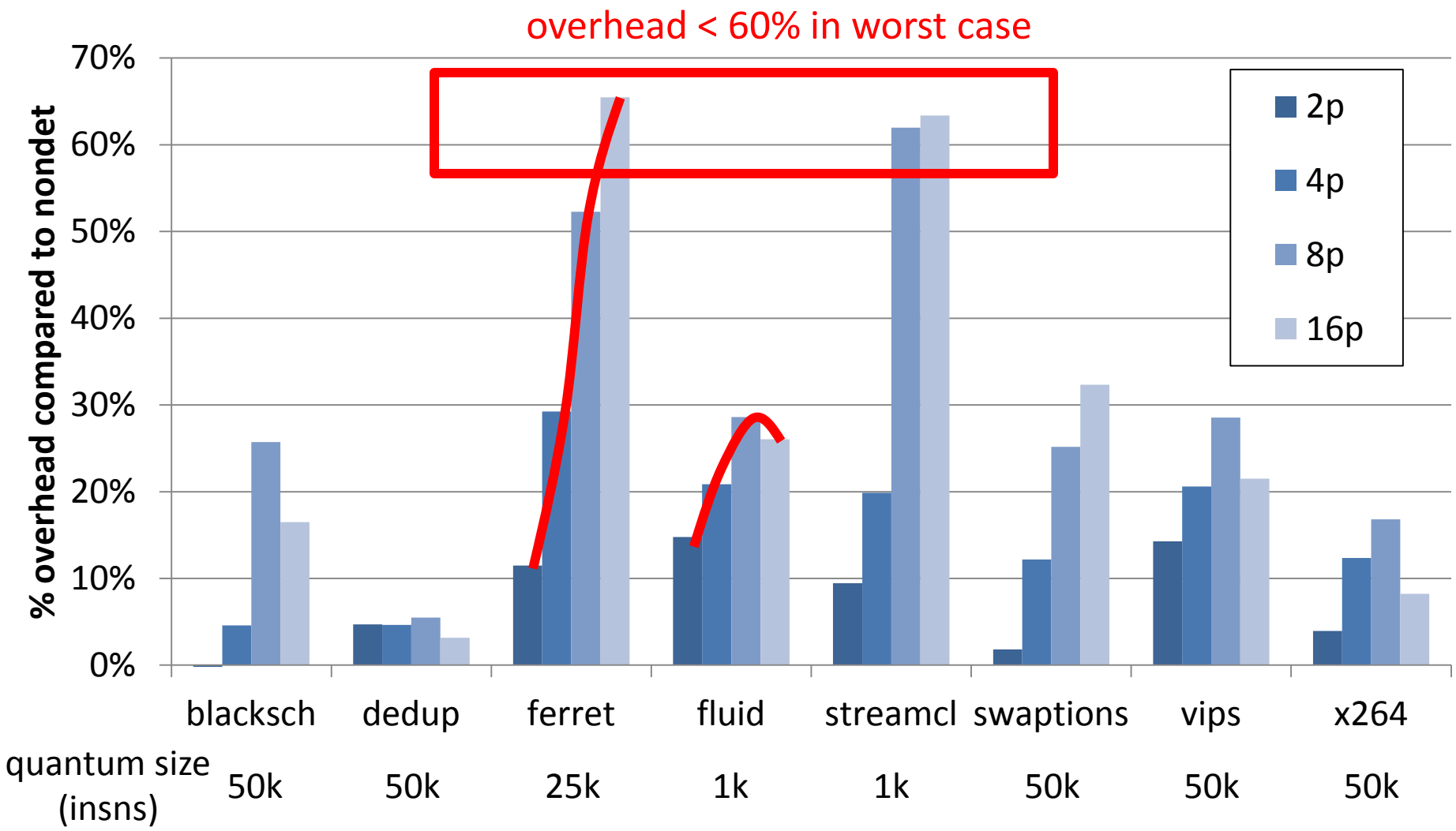
Experimental Setup

Pin-based simulator
1 IPC, except for memory ops
PARSEC v2.1 with simsmall inputs

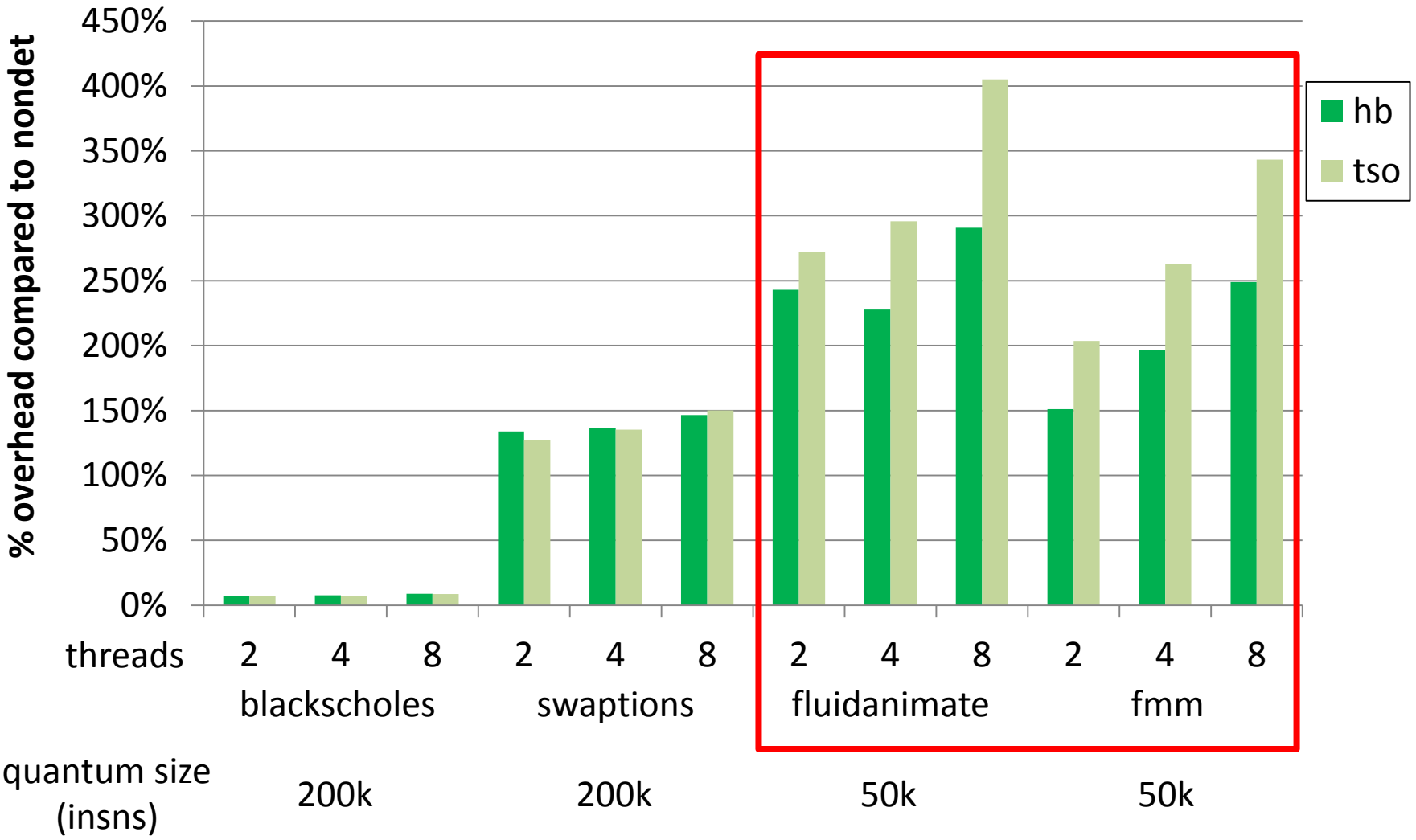
structure	size	access latency
private L1	8-way, 32KB	1 cycle
private L2	8-way, 256KB	10 cycles
shared L3	16-way, 8MB	35 cycles
memory	-	120 cycles

extended CoreDet C/C++ compiler [ASPLOS '10]
8-core Intel Harpertown @ 2.8GHz, 10GB RAM
PARSEC v2.1 with simlarge inputs

Simulation: RC/DC Overheads



Compiler: DMP-HB vs. DMP-TSO



Conclusions

- DMP-HB: a new deterministic consistency model
- RC⚡DC : a new deterministic multiprocessor design
  - no speculation
  - lightweight hardware support
- Relaxed consistency is a natural optimization for determinism

source code and data available at
http://sampa.cs.washington.edu

Thanks!

Questions?

source code and data available at
http://sampa.cs.washington.edu

DRF0 hardware requirements [ISCA '90]

1. Intra-processor dependencies are preserved.
2. All writes to the *same* location can be totally ordered based on their commit times, and this is the order in which they are observed by all processors.
3. All synchronization operations to the *same* location can be totally ordered based on their commit times, and this is also the order in which they are globally performed. Further, if *S*1 and *S*2 are synchronization operations and *S*1 is committed and globally performed before *S*2, then all components of *S*1 are committed and globally performed before any in *S*2.
4. A new access is not generated by a processor until all its previous synchronization operations (in program order) are committed.
5. Once a synchronization operation S by processor *Pi* is committed, no other synchronization operations on the *same* location by another processor can commit until after all reads of *Pi* before S (in program order) are committed and all writes of *Pi* before S are globally performed.