DeLorean: Directed Statistical Warming through Time Traveling

Best Paper Nominee

Nikos Nikoleris  Arm Research
Lieven Eeckhout  Ghent University
Erik Hagersten  Uppsala University
Trevor E. Carlson  National University of Singapore
Accuracy/Speed Trade-off

- Detailed Simulation
- Sampled Simulation
- DeLorean
- Analytical Modeling
Sampled Simulation

Practical approach: Sampled Simulation \cite{1,2}

Virtualized Fast Forwarding (KVM)

Functional Simulation: Warm-up (e.g., Caches, BP)

1% Detailed Simulation

Warm-up Warm-up D Warm-up Warm-up D

Does X hit in the cache?

Execution

[1] SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling. Wunderlich et al. ISCA 2003
[2] Full Speed Ahead: Detailed Architectural Simulation at Near-Native Speed. Sandberg et al. IISWC 2015
Sampled Simulation Bottleneck

Cache warmup
97% of simulation time
Goal: Eliminate Traditional Cache Warm-up

Feasible simulation for systems w/ large caches
But how?

Statistical Cache Modeling

Input: memory reuse

Output: miss ratio = f(cache size)

StatStack\(^1\), StatCache\(^2\), etc:
- **Sparse** input
- 20% overhead on native execution

[1] StatStack: Efficient modeling of LRU caches. Eklöv and Hagersten. ISPASS 2010.

[2] StatCache: A Probabilistic Approach to Efficient and Accurate Data Locality Analysis. Berg and Hagersten. ISPASS 2004.
DeLorean Overview

DeLorean

Simulator
- core
- DRAM

DeLorean
- Lukewarm Cache
- Associativity Model
- Capacity Model

Cache Req. ↑ Cache Resp. ↓

DRAM Req. ↓ DRAM Resp. ↑

Profile Data
- freq
- reuse dist.
The Lukewarm Cache

Do we need to statistically model every memory instruction?

Hits in the lukewarm cache are correct!

Misses in the lukewarm cache → key accesses

LRU Cache
no warm-up

© 2019 Arm Limited
Capacity Model: StatStack

Sample of reuses in the vicinity of X (RD-H_v)

Reuse of X (RD_x)

Hit=f(RD_x, RD-H_v)
Time Traveling

Record key accesses
Misses in the Lukewarm Cache

Obtain reuses for key accesses

Detailed simulation and directed cache warming

Go back in time

Go back in time

Profile

vm

ckpt

X

Y

Z

vm

ckpt

X

Y

Z

f(x)
Directed Reuse Profiling

Obtain reuses for the key accesses while executing in KVM

Watchpoint on $X \rightarrow$ Protect $P_X$

Reuse of $X$ but not the desired one

Obtain the reuse of $X$

First use

Access to $P_X$ but not to $X \rightarrow$ false positive

Avoid false positives for efficient reuse collection

© 2019 Arm Limited
Time Traveling

Record key accesses
Misses in the Lukewarm Cache

Obtain reuses key accesses

Detailed simulation and directed cache warming

Repeat until reuses for all key accesses are found.
Explorer-n+1 is not always needed!
Results: Speed

- Small number of key accesses: 49x faster than CoolSim
- Many key accesses, all Explorers needed: as fast as CoolSim
- 5.7x faster than CoolSim
- 96x faster than SMARTS

[1] CoolSim: Statistical Techniques to Replace Cache Warming with Efficient, Virtualized Profiling. Nikoleris et al. SAMOS 2016
Results: Accuracy

Average error 3.5%
Conclusion

• Multi-pass approach to identify and capture key memory reuses
  • Time Traveling: Exploits near-native fast-forwarding (KVM)
  • Targets memory reuse of key accesses

• Simulation speed
  • 96x faster than SMARTS

• Accuracy - 3% error on average

• Check the paper for more
  • Multicore simulation
  • Replacement policies

• Code publicly available at:
  • github.com/delorean-sim
Directed Statistical Warming through Time Traveling

Nikos Nikoleris
Arm Research

Lieven Eeckhout
Ghent University

Erik Hagersten
Uppsala University

Trevor E. Carlson
National University of Singapore

MICRO-52