saarland-informatics-campus.de

Warping Cache Simulation of Polyhedral Programs

Canberk Morelli, Saarland University
Jan Reineke, Saarland University

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 101020415)"

Caches appear to be important

AMD Zen 3

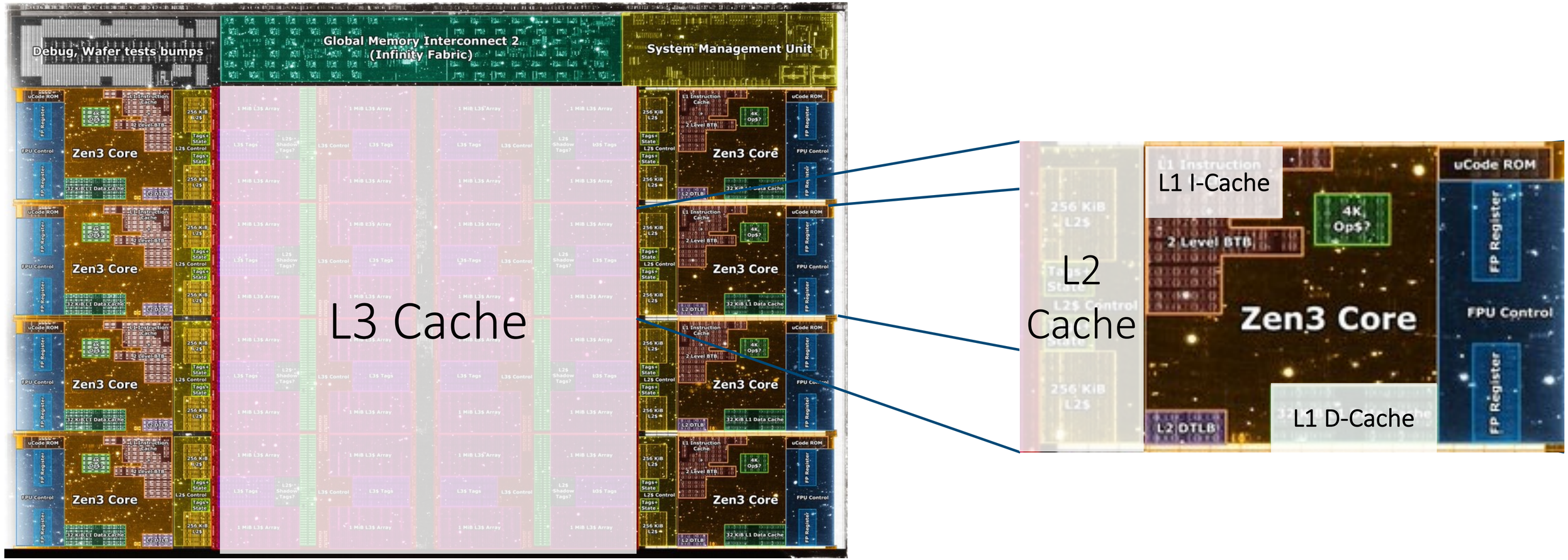


Image credit: @Locuza_ via Twitter https://twitter.com/Locuza_/status/1325534004855058432/photo/1

Cache behavior is non-obvious
Example: Matrix multiplication

```
for (int i = 0; i < 1024; i++)
  for (int j = 0; j < 1024; j++)
    for (int k = 0; k < 1024; k++)
      C[i][j] += A[i][k] * B[k][j];
```

```
for (int i = 0; i < 1024; i++)
  for (int k = 0; k < 1024; k++)
    for (int j = 0; j < 1024; j++)
      C[i][j] += A[i][k] * B[k][j];
```

L1 cache misses: 169.590.674
L2 cache misses: 166.667.689
28 seconds

L1 cache misses: 9.996.842
L2 cache misses: 397.890
5.7 seconds

... on an Intel Core i9-10980XE (Cascade Lake)

Techniques for cache analysis:
1. Trace-based cache simulators

Program

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

Program Simulator → Explicit Access Trace → Cache Simulator → # L1 Misses, # L2 Misses, …

Supports arbitrary programs + cache configurations

Analysis time is proportional to trace length

Techniques for cache analysis:
2. Analytical cache models

Program

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

=

"Polyhedral Programs"

Extract Model

Implicit Representation of Access Trace

=

Presburger Formulas

Analytical Cache Model + Counting

=

Presburger Formulas

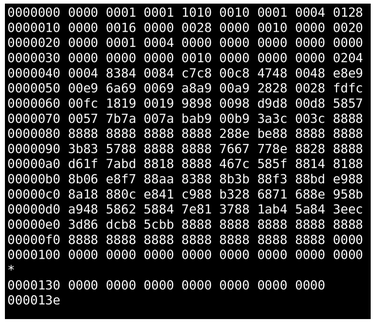
+

Barvinok's Algorithm

L1 Misses,
L2 Misses,
…

Techniques for cache analysis:
2. Analytical cache models

Program



Extract Model

Implicit Representation of Access Trace

Analytical Cache Model + Counting

L1 Misses,
L2 Misses,
…

Analysis time decoupled from trace length

Limited to restricted cache models

+ classes of programs

2. Analytical cache models: State of the art

PolyCache (Bao et al., POPL 2018)

multi-level, non-inclusive **set-associative caches** with **least-recently-used (LRU)** replacement

HayStack (Gysi et al., PLDI 2019)

multi-level, inclusive **fully-associative caches** with **least-recently-used (LRU)** replacement

[1] Bao, Krishnamoorthy, Pouchet, Sadayappan. Analytical modeling of cache behavior for affine programs. POPL 2018

[2] Gysi, Grosser, Brandner, Hoefler. A fast analytical model of fully associative caches. PLDI 2019

Real-world cache configurations

Intel Core i5-1035G1 (Ice Lake):
- L1: 48 KiB, 12-way, **LRU_3PLRU_4**
- L2: 512 KiB, 8-way, **SRRIP-HP** [*] variant
- **non-inclusive** hierarchy

AMD Zen 3:
- L1: 32 KiB, 8-way, **policy**?
- L2: 512 KiB, 8-way, **policy**?
- **inclusive** hierarchy

Intel i9-10980XE (Cascade Lake):
- L1: 32 KiB, 8-way, **Tree-PLRU**
- L2: 1 MiB, 16-way, **SRRIP-HP** [*] variant
- **non-inclusive** hierarchy

[*] Jaleel, Theobald, Steely, Emer. High Performance Cache Replacement Using Re-reference Interval Prediction (RRIP). ISCA 2010

Our goal: “Best of both worlds”

Analysis time decoupled from trace length

+

Support real-world cache configurations

Limited to restricted classes of programs

Our approach in a nutshell

Program

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

Extract Model

Implicit Representation of Access Trace

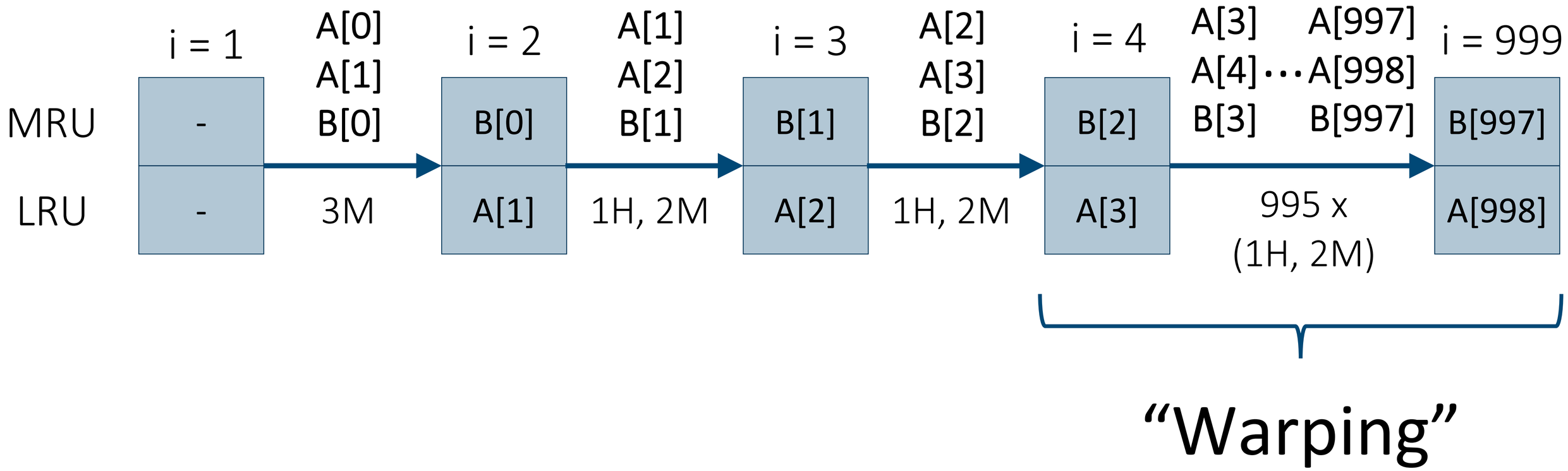
Cache Simulator

Analytical Warping

L1 Misses,
L2 Misses,
…

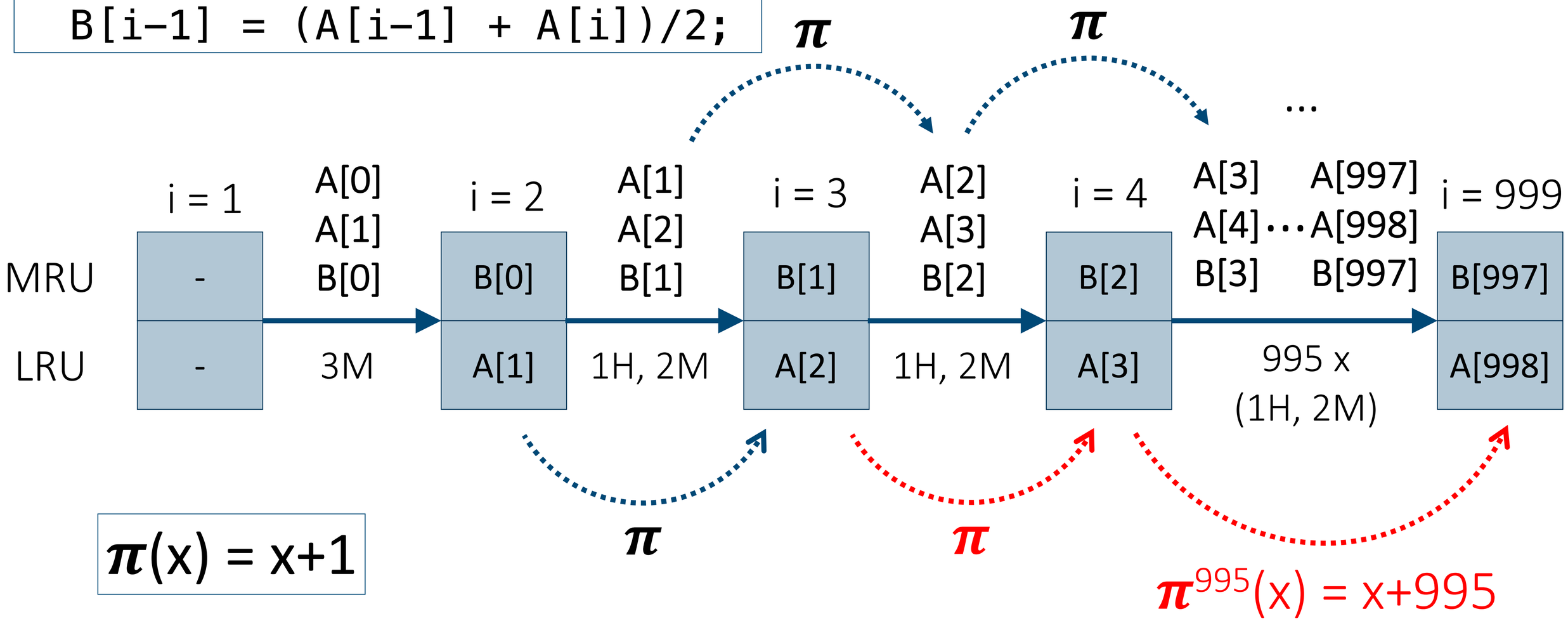
Example: 1D stencil computation

```
for (int i = 1; i < 999; i++)
  B[i-1] = (A[i-1] + A[i])/2;
```



Example revisited

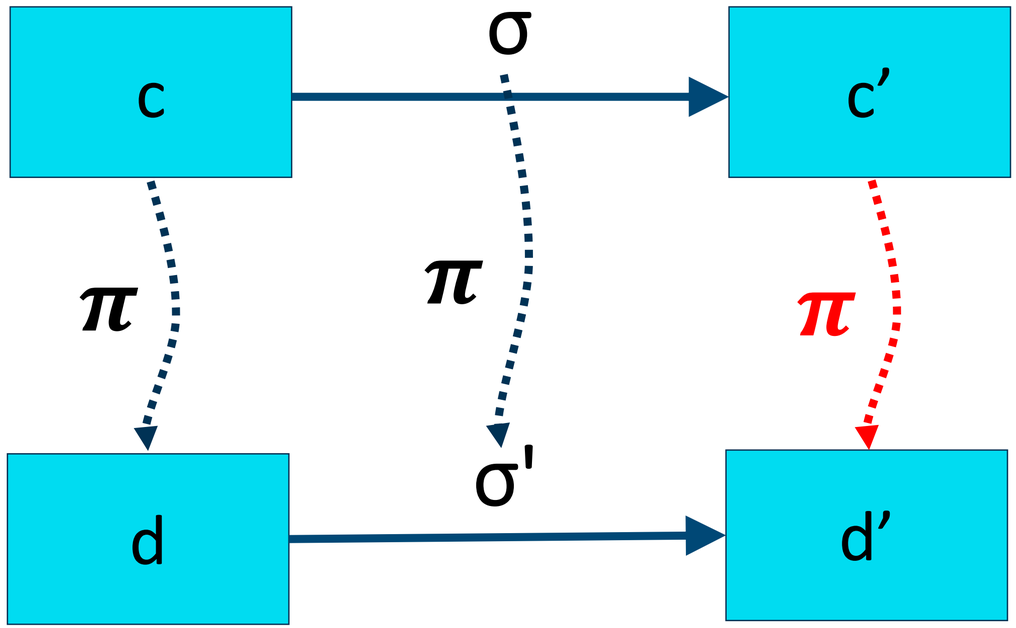
```
for (int i = 1; i < 999; i++)
  B[i-1] = (A[i-1] + A[i])/2;
```



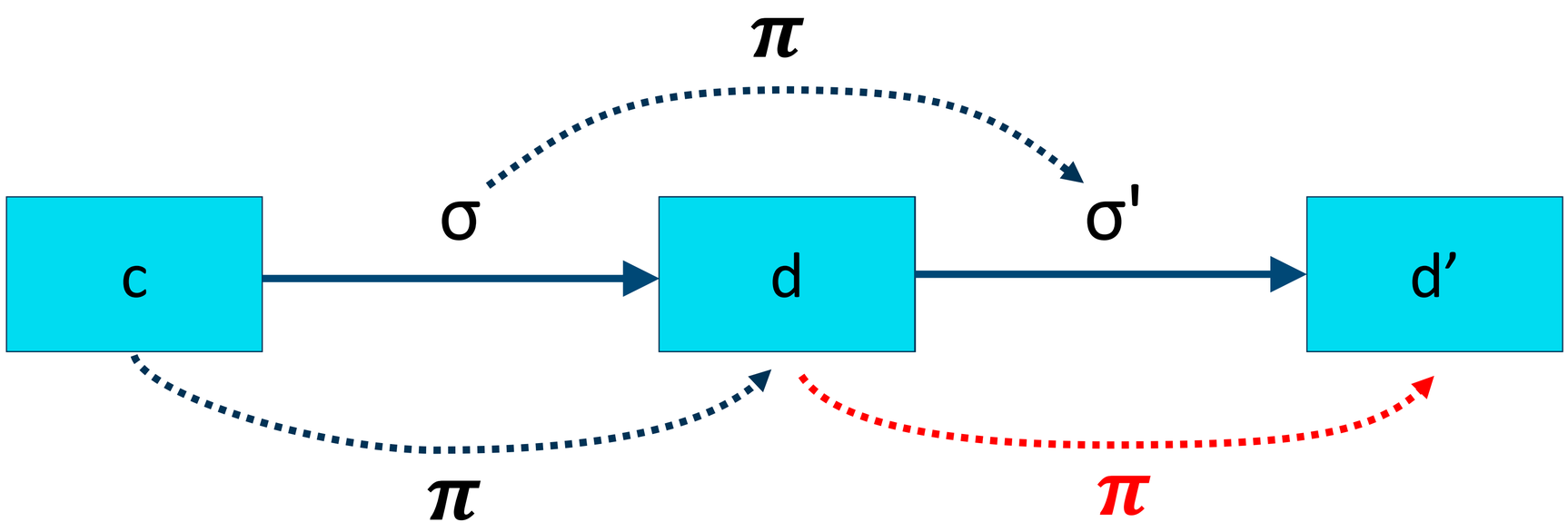
$\pi(x) = x+1$

$\pi^{995}(x) = x+995$

Key property: Data independence



Key property: Data independence



Data independence is also satisfied by

- cache replacement policies other than LRU
- set-associative caches
- cache hierarchies, e.g. L1+L2+L3

In our paper + technical report

arXiv:2203.14845v1 [cs.PL] 28 Mar 2022

Warping Cache Simulation of Polyhedral Programs*

CANBERK MORELLI, Saarland University, Germany
JAN REINEKE, Saarland University, Germany

Techniques to evaluate a program's cache performance fall into two camps: 1. Traditional trace-based cache simulators precisely account for sophisticated real-world cache models and support arbitrary workloads, but their runtime is proportional to the number of memory accesses performed by the program under analysis. 2. Relying on implicit workload characterizations such as the polyhedral model, analytical approaches often achieve problem-size-independent runtimes, but so far have been limited to idealized cache models.

We introduce a hybrid approach, warping cache simulation, that aims to achieve applicability to real-world cache models and problem-size-independent runtimes. As prior analytical approaches, we focus on programs in the polyhedral model, which allows to reason about the sequence of memory accesses analytically. Combining this analytical reasoning with information about the cache behavior obtained from explicit cache simulation allows us to soundly fast-forward the simulation. By this process of warping, we accelerate the simulation so that its cost is often independent of the number of memory accesses.

CCS Concepts: • **Software and its engineering** → **Software performance**; **Automated static analysis**.

Additional Key Words and Phrases: cache model, simulation, performance analysis, data independence

1 INTRODUCTION

Traditionally, the efficiency of an algorithm has been determined by evaluating its time complexity. Today, evaluating an algorithm's cache performance has become equally important. Over the past thirty years, the increasing processor-memory gap has led to the introduction of complex memory hierarchies consisting, in particular, of multiple cache levels. As a consequence, a program's runtime on modern hardware heavily depends on how well it exploits the underlying memory hierarchy. However, unlike time complexity, cache performance cannot easily be gauged in a compositional manner from a program's parts, i.e., the composition of two cache-efficient parts may be cache inefficient, and vice versa.

This calls for automatic methods to evaluate a program's cache performance, to inform programmers and compilers so that they can make informed choices about data-locality transformations. Cache performance analysis has already received considerable attention. Prior work can roughly be divided into two camps:

1. *Traditional cache simulators*, such as Dinero IV [20] or CASPER [38], simulate a program's cache behavior by explicitly iterating over the trace of memory accesses generated by the program. The advantage of this approach is that it is applicable to arbitrary workloads and it is possible to precisely model modern memory hierarchies, including sophisticated cache replacement policies, such as Pseudo-LRU [3] or Quad-age LRU [39, 40] found in real-world microarchitectures [2, 65]. The main drawback of traditional simulators is that their runtime is *proportional to the number of* accesses a program performs. As a consequence, the simulation of programs operating on ... may take weeks or more.
... 11, 14, 15, 25, 26, 34, 59, 60], on the other hand, e.g. PolyCache [7] ... is times that are *independent of the number of memory* ... To this end, they rely on implicit representations ... m representation is the *polyhedral*

- Data independence of
  - Set-associative caches
  - Hierarchical caches, e.g. L1+L2+L3
- Symbolic simulation + hashing to efficiently detect "matches"
- Checking necessary conditions via polyhedral techniques

Experimental evaluation

Experimental evaluation

Performance: Is warping effective?

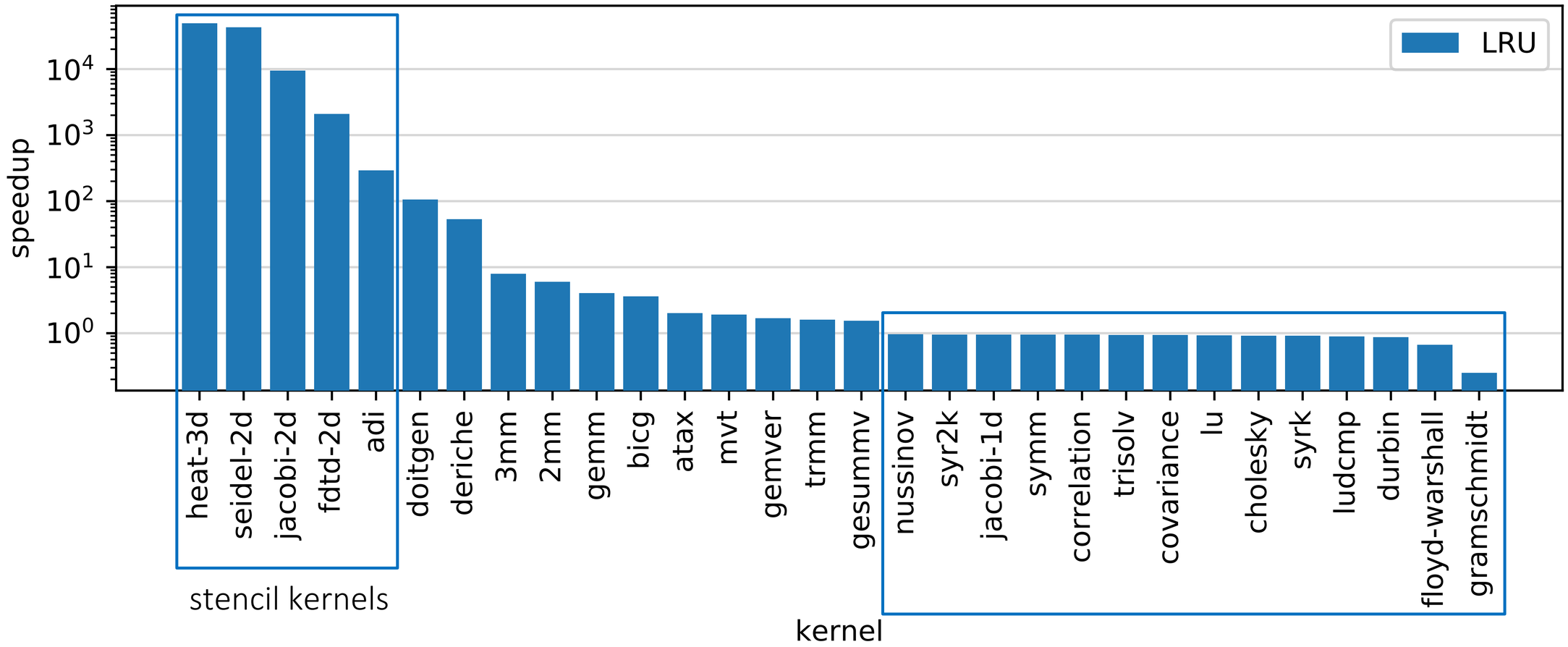
Does it matter to accurately model real-world caches?

Performance: Speedup due to warping

SIC Saarland Informatics Campus

PolyBench - problem size L

8-way 32 KiB L1 cache under LRU replacement

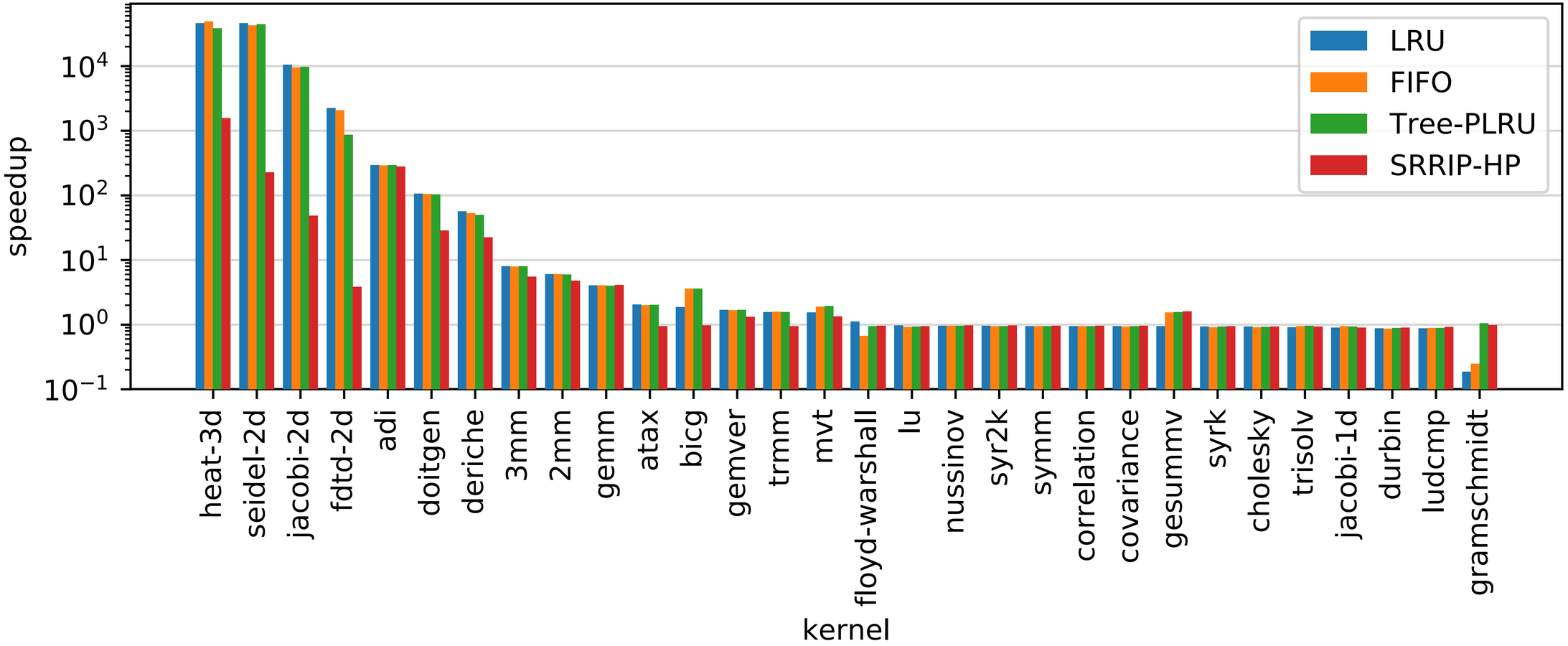


Performance: Speedup due to warping

SIC Saarland Informatics Campus

PolyBench - problem size L

8-way 32 KiB L1 cache under LRU, FIFO, Tree-PLRU, SRRIP-HP

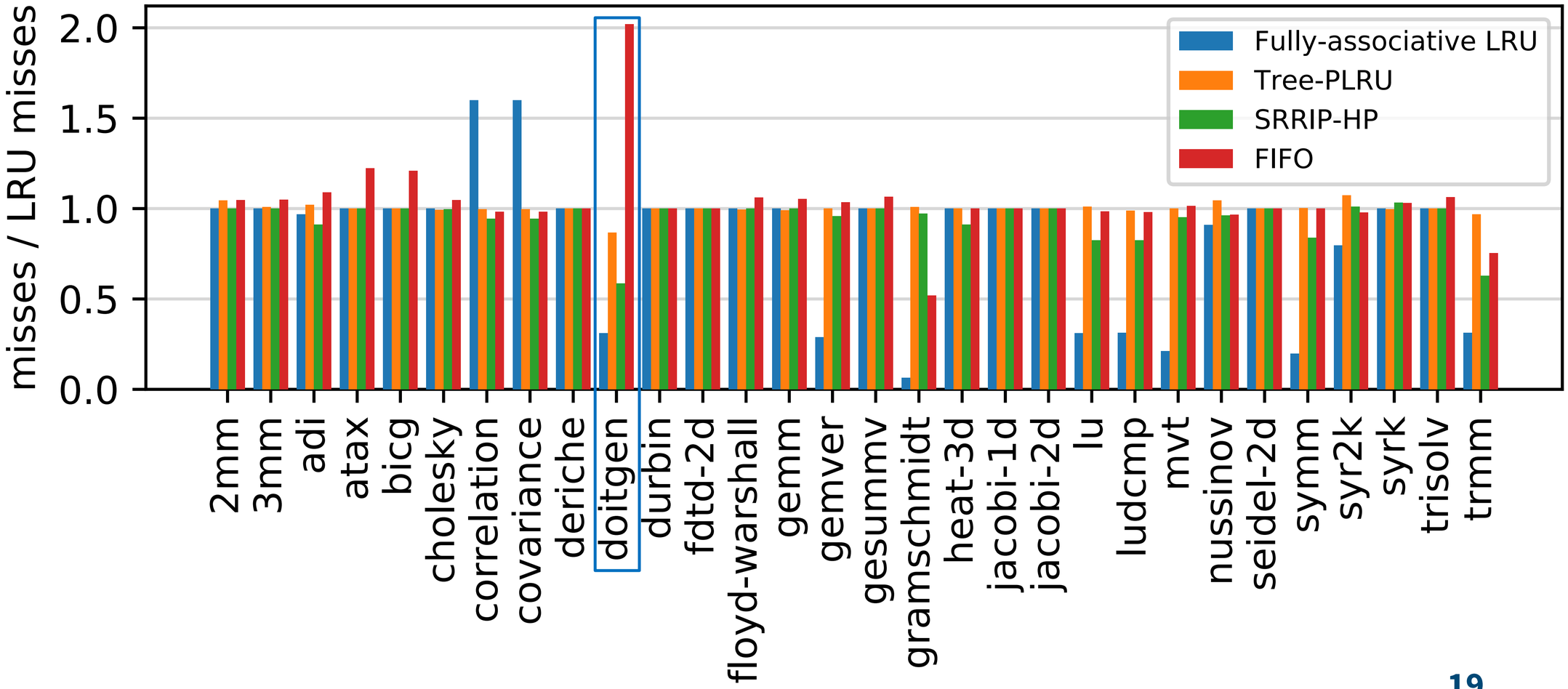


Does it matter to model real-world caches?

SIC Saarland Informatics Campus

PolyBench - problem size M

Fully-associative LRU, Tree-PLRU, SRRIP-HP, FIFO relative to set-associative LRU



The End

SIC Saarland Informatics Campus

Program

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

Extract Model

Implicit Representation of Access Trace

Concrete Cache Simulator

Generic Analytical Warping

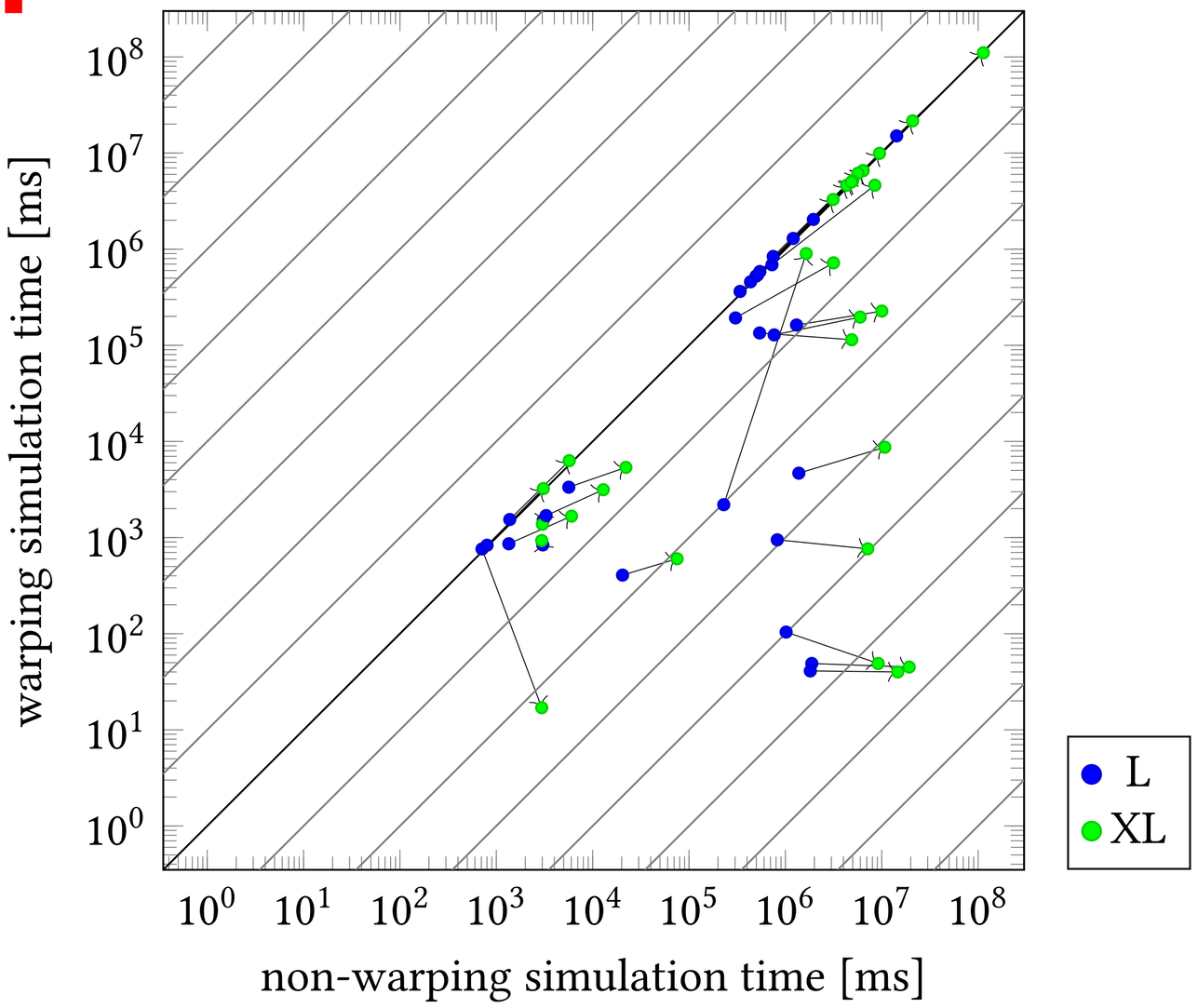
L1 Misses,
L2 Misses,
…

Questions?

Backup Slides

Warping vs non-warping simulation
Scaling behavior

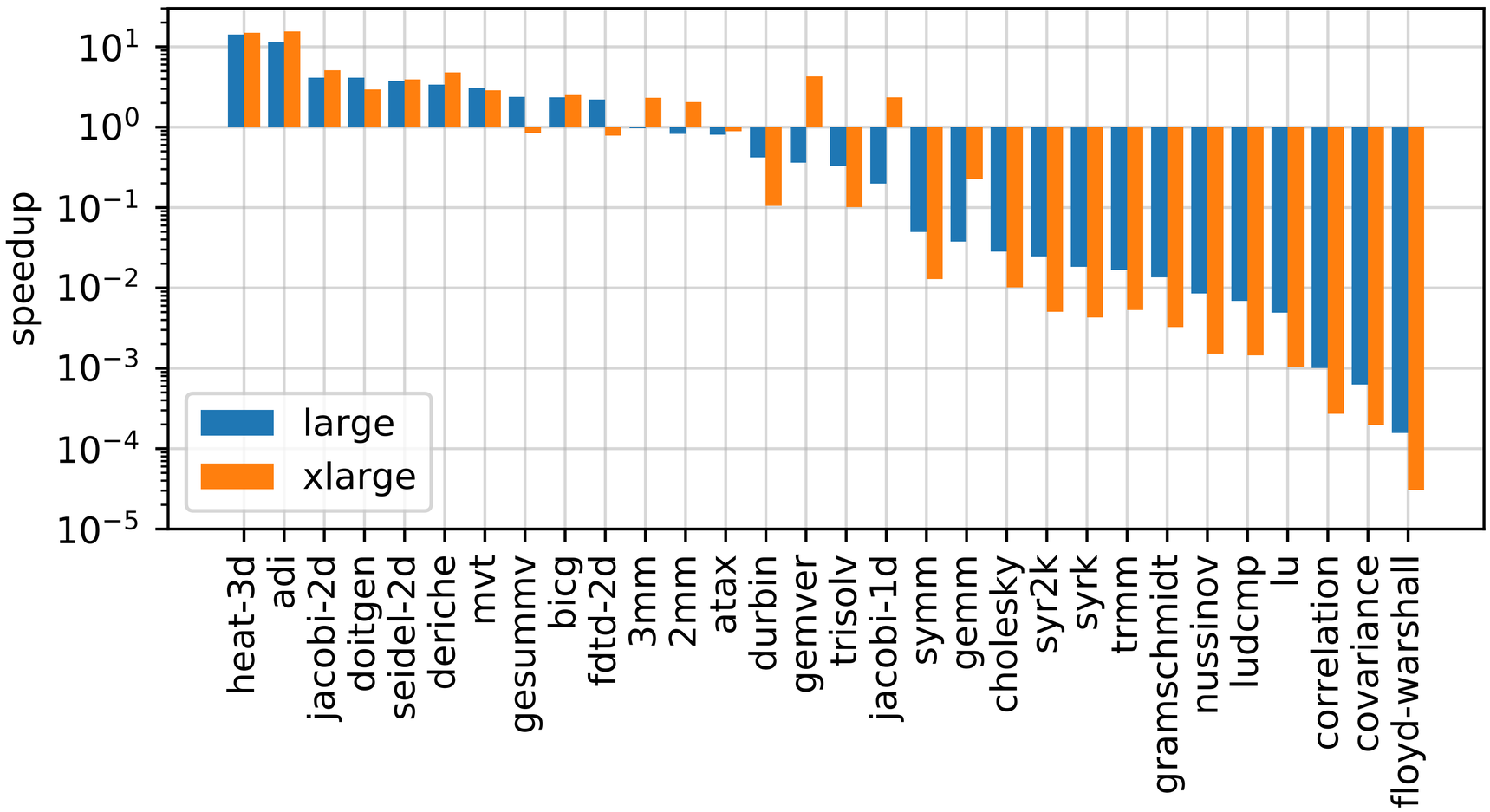
PolyBench -
problem size (L)
vs problem size (XL)



Warping vs HayStack

PolyBench -
problem size L+XL

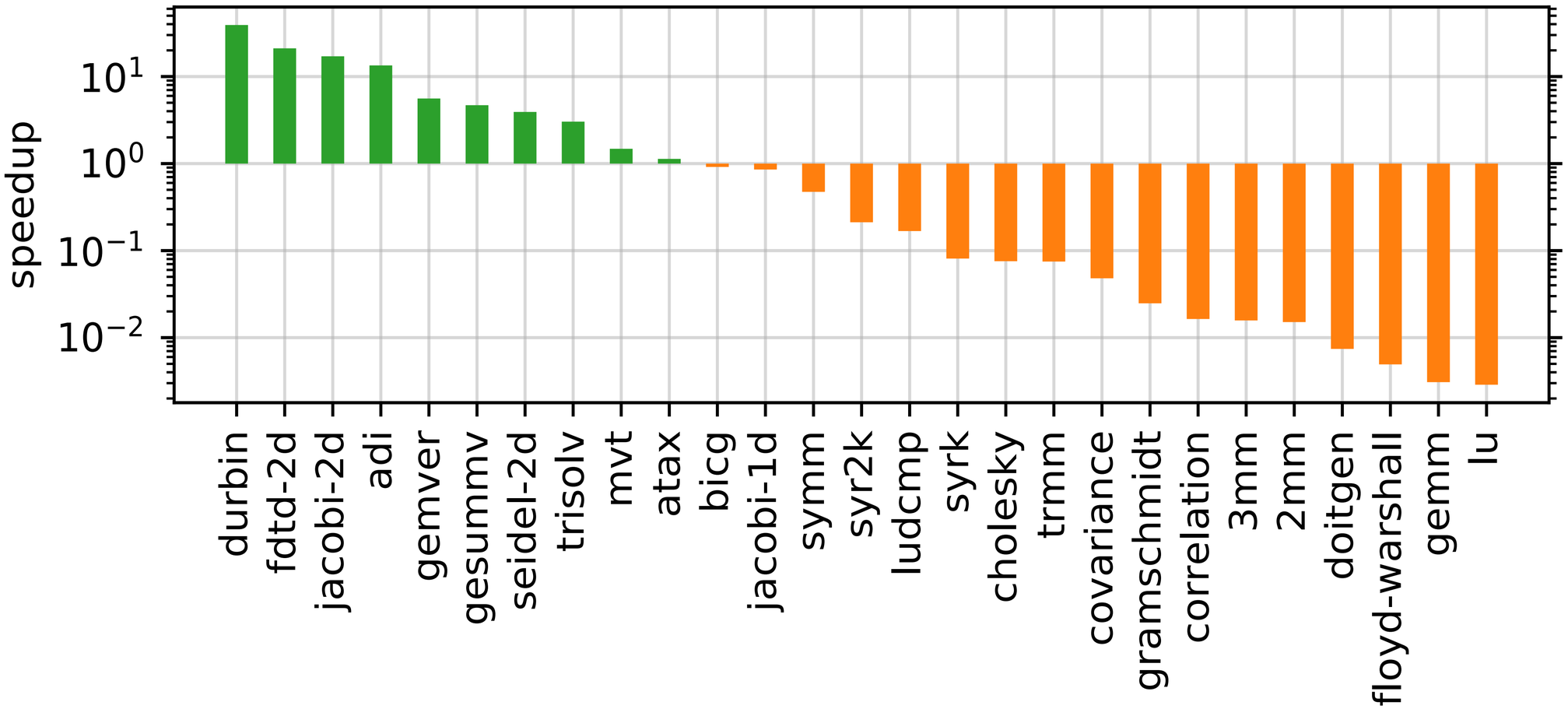
32 KiB fully-associative LRU



Warping vs PolyCache

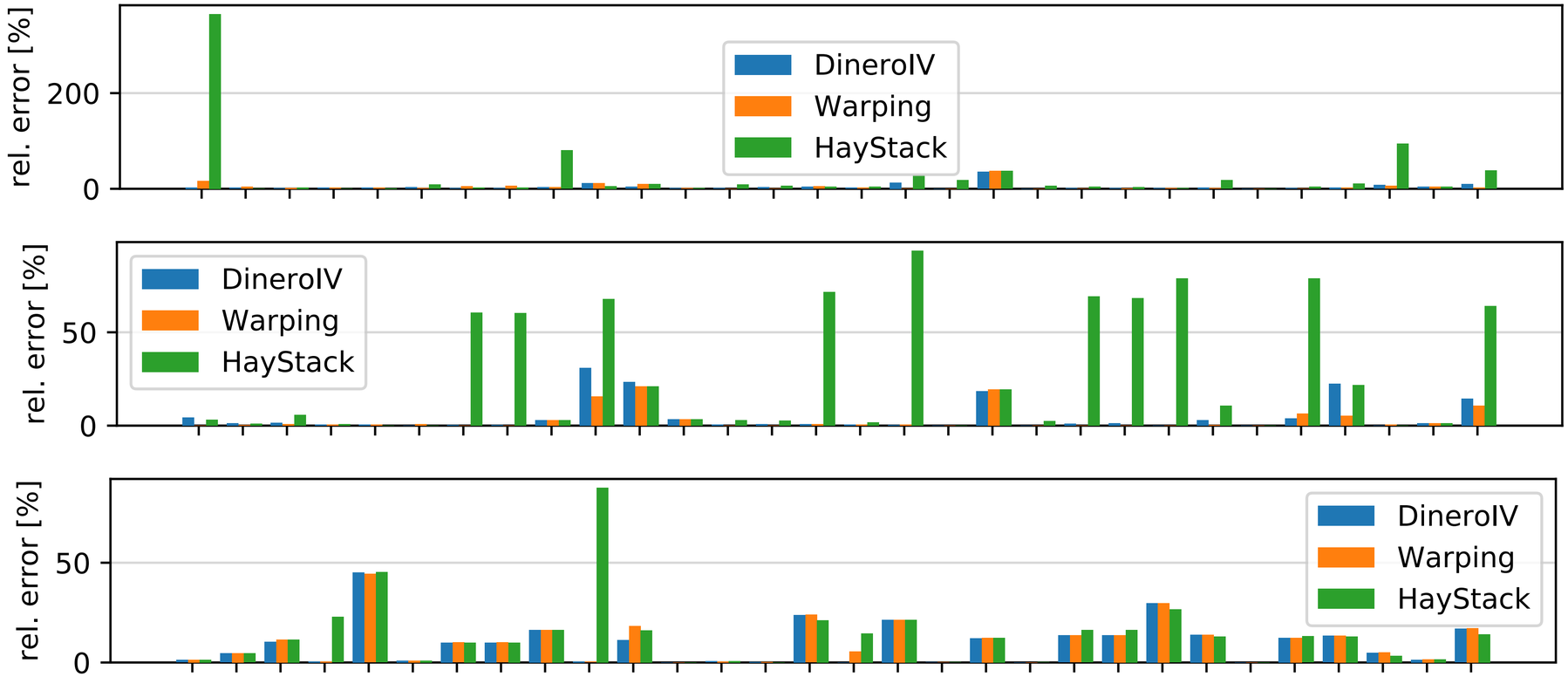
PolyBench - problem size L

L1: 32 KiB 4-way set-associative LRU
L2: 256 KiB 4-way set-associative LRU



Accuracy relative to measurements

System: Intel i9-10980XE (Cascade Lake) with PLRU replacement
Measurements using PAPI



problem size:

“small”

“medium”

“large”