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];
```

L1 cache misses: 169,590,674
L2 cache misses: 166,667,689
28 seconds

... on an Intel Core i9-10980XE (Cascade Lake)
Techniques for cache analysis:  
1. Trace-based cache simulators

- Supports arbitrary programs + cache configurations
- Analysis time is proportional to trace length
Techniques for cache analysis:

2. Analytical cache models

Program

= “Polyhedral Programs”

Extract Model

= Implicit Representation of Access Trace

Analytical Cache Model + Counting

= Presburger Formulas + Barvinok’s Algorithm

= # L1 Misses, # L2 Misses, ...
Techniques for cache analysis:

2. Analytical cache models

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 \textbf{set-associative caches} with
least-recently-used (LRU) replacement

HayStack (Gysi et al., PLDI 2019)
multi-level, inclusive \textbf{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, *LRU₃PLRU₄*
- 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 → Extract Model → Implicit Representation of Access Trace → Cache Simulator → Analytical Warping → # L1 Misses, # L2 Misses, ...
Example: 1D stencil computation

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

**"Warping"**
Example revisited

\[
\text{for (int } i = 1; i < 999; i++) \\
B[i-1] = (A[i-1] + A[i])/2;
\]

\[
\pi(x) = x + 1
\]

\[
\pi^{995}(x) = x + 995
\]
Key property: Data independence
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

- 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

PolyBench - problem size L

8-way 32 KiB L1 cache under LRU replacement
Performance: Speedup due to warping

PolyBench - problem size L

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

![Graph showing speedup for various kernels under different cache policies.](image-url)

- LRU
- FIFO
- Tree-PLRU
- SRRIP-HP

kernel:
- heat3d
- seidel-2d
- jacobi-2d
- ftdt-2d
- adi
- doitgen
- deriche
- 3mm
- 2mm
- gemm
- atax
- bicg
- gemver
- trmm
- mvt
- lu
- nussinov
- syr2k
- symm
- correlation
- covariance
- gesummv
- syrk
- cholesky
- trisolv
- jacobi-1d
- durbin
- ludcmp
- gramschmidt

speedup:
Does it matter to model real-world caches?

PolyBench - problem size M

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

Questions?
Backup Slides
Warping vs non-warping simulation

Scaling behavior

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

Fig. 7. L1 warping and non-warping simulation times for problem sizes L and XL.

6.2 Warping vs Non-Warping Simulation

Warping vs non-warping simulation. We first simulate the L1 cache of the test system for problem size L. To investigate the effect of the replacement policy on the warping performance, in addition to the Pseudo-LRU policy of the test system, we also simulate LRU, FIFO, and Quad-age LRU.

Figure 6 shows for each benchmark the speedup of warping simulation compared to the non-warping simulation (bottom) and the share of non-warped accesses (top). The reported times correspond to the time spent executing the implementations of Algorithms 1 and 2, i.e., they do not include the overhead of extracting the internal representation of the benchmarks via isl. This overhead, which is identical for warping and non-warping simulation, lies between 62 and 245 ms depending on the benchmark and is thus dominated by the simulation time for most benchmarks.

The first observation is that the speedup is roughly inversely proportional to the share of non-warped accesses. For, e.g., adi, about 0.3% of all accesses cannot be warped and we observe a speedup of about $300^G$.

The stencil kernels adi, fdtd-2d, heat-3d, jacobi-2d, and seidel-2d exhibit large speedups. Stencils have uniformly generated references, and thus give rise to recurring patterns in the cache if there are enough accesses relative to the cache size. As we discussed earlier, warping aims to exploit these patterns to accelerate the simulation. The consistent speedups for the stencil kernels show that warping simulation is indeed able to achieve this. The jacobi-1d kernel does not benefit from warping since its working set is too small to fill the cache.

While there are many kernels that benefit from warping, there are others that do not. We observed that there were no (or very few) symbolically equivalent cache states during the simulation of these kernels, and thus, no (or very few) opportunities for warping. As we show later, some of these kernels benefit from warping when simulating a different cache. However, for the current cache configuration, we conclude that warping does not decrease the simulation times of these kernels.

Overall, the differences between the replacement policies are fairly small, with LRU, Pseudo-LRU, and FIFO often exhibiting similar speedups. Quad-age LRU is scan- and thrash-resistant, which may result in “old” memory blocks remaining in the cache, while scanning through new ones, which in some cases results in a greater number of classic simulation steps before detecting warping opportunities.
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”