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

```c
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)

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

= Presburger Formulas + Barvinok’s Algorithm

# L1 Misses, # L2 Misses, ...

Program

= Presburger Formulas

Analytical Cache Model + Counting
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 *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
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

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;
```

```
<table>
<thead>
<tr>
<th>i = 1</th>
<th>i = 2</th>
<th>i = 3</th>
<th>i = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>3M</td>
<td>1H, 2M</td>
<td>1H, 2M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1H, 2M</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>995 x (1H, 2M)</td>
<td></td>
</tr>
</tbody>
</table>
```

“Warping”
Example revisited

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

\[
\pi^{995}(x) = x + 995
\]

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

- 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

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
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
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

speedup

large
xlarge
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”