Abstract Interpretation of FIFO Replacement

Daniel Grund    Jan Reineke

Saarland University, Saarbrücken, Germany

Static Analysis Symposium 2009
Outline

1 Introduction & Motivation
   - Timing Analysis
   - Cache Analysis

2 Abstract Interpretation of FIFO
   - Challenge FIFO Replacement
   - Domain Cooperation
   - Must Analysis
   - May Analysis

3 Evaluation
   - Related Work
   - Analysis Precision

4 Summary
Outline

1  Introduction & Motivation
   ■ Timing Analysis
   ■ Cache Analysis

2  Abstract Interpretation of FIFO
   ■ Challenge FIFO Replacement
   ■ Domain Cooperation
   ■ Must Analysis
   ■ May Analysis

3  Evaluation
   ■ Related Work
   ■ Analysis Precision

4  Summary
Notions in Timing Analysis

- Execution time depends on
  - program input
  - initial hardware state

- Bounds required for schedulability analysis of real-time systems
Static Timing-Analysis Framework

Framework implemented by aiT of AbsInt

Micro-architectural analysis

- models pipeline, caches, buses, etc.
- derives bounds on BB exec. times
- is an abstract interpretation with a huge domain
- is the computationally most expensive module
Caches and Replacement Policies

- Caches transparently buffer memory blocks
Caches and Replacement Policies

- Caches transparently buffer memory blocks
Caches and Replacement Policies

- Caches transparently buffer memory blocks
Caches and Replacement Policies

- Caches transparently buffer memory blocks

CPU → "miss" [ab] → Main Memory

Capacity: 32 KB
Latency: 3 cycles

Capacity: 2 MB
Latency: 100 cycles
Caches and Replacement Policies

- Caches transparently buffer memory blocks
Caches and Replacement Policies

- Caches transparently buffer memory blocks.
- Replacement policy *dynamically* decides which element to replace:
  - LRU  least recently used
  - PLRU pseudo LRU
  - FIFO  first-in first-out
- Have great influence on abstraction and (obtainable) analysis precision.
Caches and Replacement Policies

- Caches transparently buffer memory blocks
- Replacement policy *dynamically* decides which element to replace
  - LRU  least recently used
  - PLRU  pseudo LRU
  - FIFO  first-in first-out
- Have great influence on abstraction and (obtainable) analysis precision
Cache Analysis: Motivation & Application

- Cache performance has great influence on overall performance
- Need tight bounds on cache performance
- Otherwise derived timing bounds may be useless:
  - tasks are deemed not schedulable
  - waste of hardware resources

- Application: Buffers with transparent replacement
  - Instruction- and data-caches
  - Branch target buffers (BTB, BTIC)
  - Translation lookaside buffers (TLB)
Static Cache Analysis

- derives approximations to cache contents at each program point
- in order to classify memory accesses as cache hits or cache misses

<table>
<thead>
<tr>
<th>Must-information</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Underapproximation of cache contents</td>
</tr>
<tr>
<td>- Used to soundly classify cache hits</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>May-information</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Overapproximation of cache contents</td>
</tr>
<tr>
<td>- Used to soundly classify cache misses</td>
</tr>
</tbody>
</table>
Outline

1 Introduction & Motivation
   ■ Timing Analysis
   ■ Cache Analysis

2 Abstract Interpretation of FIFO
   ■ Challenge FIFO Replacement
   ■ Domain Cooperation
   ■ Must Analysis
   ■ May Analysis

3 Evaluation
   ■ Related Work
   ■ Analysis Precision

4 Summary
Concrete Semantics: What is FIFO?

- State of FIFO cache of size $k$: $s \in S := T^k$
  
  \[
  \begin{array}{c}
  \text{last-in} \\
  \downarrow \\
  [t_0, \ldots, t_{k-1}]
  \end{array}
  \quad \begin{array}{c}
  \text{first-in} \\
  \downarrow \\
  [t_0, \ldots, t_{k-1}]
  \end{array}
  \]

- Examples:
  
  \[
  [d, c, b, a] \xrightarrow{c_{hit}} [d, c, b, a]
  \]

  \[
  [d, c, b, a] \xrightarrow{e_{miss}} [e, d, c, b]
  \]

- Update: $U_S : S \times T \rightarrow S$

  \[
  U_S([t_0, \ldots, t_{k-1}], t) := \begin{cases}
  [t_0, \ldots, t_{k-1}] & : \exists i : t = t_i \quad \text{“cache hit”} \\
  [t, t_0, \ldots, t_{k-2}] & : \text{otherwise} \quad \text{“cache miss”}
  \end{cases}
  \]
Challenge: How to Predict Hits?

- Consider a FIFO cache with unknown contents

\[
\begin{align*}
[?, ?, ?, ?] & \xrightarrow{a\text{ hit}} [?, a, ?, ?] \xrightarrow{b\text{ hit}} [?, a, ?, b] \\
[?, ?, ?, ?] & \xrightarrow{a\text{ hit}} [?, ?, ?, a] \xrightarrow{b\text{ miss}} [b, ?, ?, ?]
\end{align*}
\]

- If \( a \) may be a hit, then \( b \) may evict \( a \)
  \[\Rightarrow\] Can only predict hits for most recently accessed element

- Can one do better?

\[
\begin{align*}
[?, ?, ?, ?] & \xrightarrow{a\text{ miss}} [a, ?, ?, ?] \xrightarrow{b\text{ miss}} [b, a, ?, ?]
\end{align*}
\]

- If \( a \) is a miss, then \( b \) cannot evict \( a \)
  \[\Rightarrow\] Can predict hits for \( a \) until \( k \) further misses might have happened
Challenge: How to Predict Hits?

- Consider a FIFO cache with unknown contents

\[
\begin{align*}
[?, ?, ?, ?] & \xrightarrow{a \text{ hit}} [?, a, ?, ?] \xrightarrow{b \text{ hit}} [?, a, ?, b] \\
[?, ?, ?, ?] & \xrightarrow{a \text{ hit}} [?, ?, ?, a] \xrightarrow{b \text{ miss}} [b, ?, ?, ?]
\end{align*}
\]

- If \( a \) may be a hit, then \( b \) may evict \( a \)

\[ \Rightarrow \text{ Can only predict hits for most recently accessed element} \]

- Can one do better?

\[
[?, ?, ?, ?] \xrightarrow{a \text{ miss}} [a, ?, ?, ?] \xrightarrow{b \text{ miss}} [b, a, ?, ?]
\]

- If \( a \) is a miss, then \( b \) cannot evict \( a \)

\[ \Rightarrow \text{ Can predict hits for } a \text{ until } k \text{ further misses might have happened} \]

\[ \Rightarrow \text{ Need may-information to obtain precise must-information} \]
A Solution. Well, our Contributions

- Framework for static cache analysis
  - policy independent
  - couples must- and may-analyses
  - analyses cooperate via “update reduction”

- FIFO must-analysis
  - can profit from may-information
  - hence, also better must-information

- FIFO may-analysis
  - utilizes order of hits and misses
  - more precise than prior analyses
Framework and Classification

Domain: \( \text{Fifo} := \text{Must} \times \text{May} \)

Classification: \( \text{Class} := \{\text{H, M}\}^{\top} \)

\( \begin{array}{c}
\text{H} : \text{cache hit} \\
\text{M} : \text{cache miss} \\
\text{T} : \text{unclassified}
\end{array} \)

\( C_{\text{Fifo}} : \text{Fifo} \times \mathcal{T} \rightarrow \text{Class} \)

\( C_{\text{Fifo}}((\text{must}, \text{may}), t) := C_{\text{Must}}(\text{must}, t) \sqcap C_{\text{May}}(\text{may}, t) \)
Domain Cooperation via Update Reduction

\[(must, may) \quad \text{Independent} \quad \rightarrow \quad (must', may')\]
Domain Cooperation via Update Reduction

\[(\text{must}, \text{may}) \xrightarrow{\text{Independent}} (\text{must}', \text{may}')\]

\[\gamma \xrightarrow{\mathcal{U}_S} S \xrightarrow{\alpha} S\]
Domain Cooperation via Update Reduction

\[(must, \ may)\]

Independent
is imprecise

\[(must', \ may')\]

\[\gamma \quad \gamma\]

\[\alpha \quad \alpha\]

\[\mathcal{U}_S\]

\[S\]

\[S\]
Domain Cooperation via Update Reduction

Best possible

$$(must, may) \rightarrow (must', may')$$
Domain Cooperation via Update Reduction

Best possible

\[(must, may) \rightarrow (must', may')\]
Domain Cooperation via Update Reduction

\[(must, may) \xrightarrow{\text{Best possible}} (must', may')\]
Domain Cooperation via Update Reduction

Best possible is infeasible

\((must, may)\) \(\xrightarrow{\gamma} \gamma \xrightarrow{\gamma} \alpha\) \(\xrightarrow{\alpha} \) \((must', may')\)
Domain Cooperation via Update Reduction

\[
\begin{align*}
(must, may) & \xrightarrow{\text{Using classification}} \\
C_{\text{Fifo}}((must, may), t) & = M \\
\rightarrow (must', may')
\end{align*}
\]
Domain Cooperation via Update Reduction

\[(must, may) \xrightarrow{C_{Fifo}((must, may), t) = M} (must', may')\]
Domain Cooperation via Update Reduction

\[(must, may) \xrightarrow{\text{Using classification}} C_{\text{FIFO}}((must, may), t) = M \rightarrow (must', may')\]

\[\gamma_M(t) = \{s \in S \mid t \notin s\}\]

\[S\]
Domain Cooperation via Update Reduction

\[ (\text{must, may}) \xrightarrow{\text{Using classification}} C_{\text{Fifo}}((\text{must, may}), t) = M \xrightarrow{} (\text{must}', \text{may}') \]

\[ \gamma \quad \alpha \]

\[ \gamma_M(t) = \{ s \in S \mid t \notin s \} \]

\[ \mathcal{U}_S \]
Outline

1 Introduction & Motivation
   - Timing Analysis
   - Cache Analysis

2 Abstract Interpretation of FIFO
   - Challenge FIFO Replacement
   - Domain Cooperation
   - Must Analysis
   - May Analysis

3 Evaluation
   - Related Work
   - Analysis Precision

4 Summary
Must-Analysis: Potential Misses

- For FIFO, a newly inserted element is evicted after \( k \) misses

\[ \Rightarrow \text{Maintain upper bound on number of misses: Potential misses} \]

- Abstract must-domain closely resembles the concrete domain

\[ \text{Must}_{Fifo_k} := [T_0, \ldots, T_{k-1}] , \]

where \( T_i \in \mathcal{P}(T) \), \( T_i \cap T_j = \emptyset \), and \( \sum |T_i| \leq k \).

- \( t \in T_i \Rightarrow \) at most \( i \) misses since insertion of \( t \)

- Concretization example

\[ \triangleright (\{\{f\}, \emptyset, \{a, c\}, \{b\}\}) \equiv \{[f, c, a, b], [f, a, c, b]\} \]
Must-Analysis: Update

\[ \mathcal{U}_{\text{Must}} : \text{Must} \times \mathcal{T} \times \text{Class} \rightarrow \text{Must} \]

\[ \mathcal{U}_{\text{Must}}([T_0, \ldots, T_{k-1}], t, cl) := \begin{cases} 
[\emptyset, T_0, \ldots, T_{k-2} \cup \{t\}] & : cl \equiv \top \\
[T_0, \ldots, T_{k-1}] & : cl \equiv \bot \\
\{t\}, T_0, \ldots, T_{k-2} & : cl \equiv M 
\end{cases} \]

- **Misses** classified by may-analysis
- **Last case** only possible due to **domain cooperation**
Outline

1 Introduction & Motivation
   - Timing Analysis
   - Cache Analysis

2 Abstract Interpretation of FIFO
   - Challenge FIFO Replacement
   - Domain Cooperation
   - Must Analysis
   - May Analysis

3 Evaluation
   - Related Work
   - Analysis Precision

4 Summary
May-Analysis: Definite Misses

- How to predict misses?
- Maintain lower bound on number of misses: Definite misses

- Initially, anything might be cached
- To classify a miss for an individual access, one needs to predict $k$ other misses first

**Lemma**

A newly inserted element is evicted after accesses to at most $2k < 1$ pairwise different elements.
May-Analysis: Idea “Early Misses”

- Initial state: \([x, c, b, a]\)
- Sequences of different length:
  - \(\langle a, b, c, e, f, g, h \rangle\)
- Common final state: \([h, g, f, e]\)
May-Analysis: Idea “Early Misses”

- Initial state: \([x, c, b, a]\)
- Sequences of different length:
  - \([a, b, c, e, f, g, h]\)
  - \([e, f, g, h]\)
- Common final state: \([h, g, f, e]\)
May-Analysis: Idea “Early Misses”

- Initial state: \([x, c, b, a]\)
- Sequences of different length:
  - \([a, b, c, e, f, g, h]\)
  - \([e, f, g, h]\)
  - \([a, e, f, c, g, h]\)
- Common final state: \([h, g, f, e]\)
May-Analysis: Idea “Early Misses”

- Initial state: \([x, c, b, a]\)
- Sequences of different length:
  - \([a, b, c, e, f, g, h]\)
  - \([e, f, g, h]\)
  - \([a, e, f, c, g, h]\)
- Common final state: \([h, g, f, e]\)
  \(\Rightarrow\) Sequences differ in number of hits
May-Analysis: Idea “Early Misses”

- Initial state: \([x, c, b, a]\)
- Sequences of different length:
  - \([a, b, c, e, f, g, h]\)
  - \([e, f, g, h]\)
  - \([a, e, f, c, g, h]\)
- Common final state: \([h, g, f, e]\)
  ⇒ Sequences differ in number of hits

- “Early misses”
  - preclude hits to thereby evicted elements
  - reduce number of possible accesses between insertion and eviction
  ⇒ Order of hits and misses is important
May-Analysis: Domain

- May analysis approximates position in triangle
- Unclassified access → “take the longer way”

For each element $t$, the analysis maintains:

- $A$ Set of Potentially Accessed Elements
- $dm$ Number of Definite Misses
- $cw$ Number of Covered Ways (Covered Cache Positions)
May-Analysis: Example

- Assume sequence $\langle x, a, b, c \rangle$ and all accesses are unclassified
- Then for $x$ one has:
  $A = \{a, b, c\}$  $a$, $b$ and $c$ might have been accessed since the last insertion of $x$
  $dm = 0$  0 misses have definitely happened since the last insertion of $x$
  $cw = 3$  Assuming that all unclassified accesses were hits, then 3 elements of $A$ must be cached
- Consider next access to $d$
Outline

1. Introduction & Motivation
   - Timing Analysis
   - Cache Analysis

2. Abstract Interpretation of FIFO
   - Challenge FIFO Replacement
   - Domain Cooperation
   - Must Analysis
   - May Analysis

3. Evaluation
   - Related Work
   - Analysis Precision

4. Summary
Relative Competitiveness

- How many misses would FIFO have if LRU has $m_{LRU}$ misses?

**Definition: Relative Competitiveness**

Policy $P$ is $(f, c)$ miss-competitive relative to policy $Q$ if

$$m_P(p, s) \leq f \cdot m_Q(q, s) + c$$

for all access sequences $s$ and compatible cache states $p, q$.

- E.g. LRU$(2k - 1)$ is $(1, 0)$ miss-competitive vs. FIFO$(k)$
  \[\Rightarrow\] LRU$(2k - 1)$ may-analysis can be used for FIFO$(k)$ may analysis
Evaluation Setup

- **Must-analysis:**
  - **CM** Canonical must-analysis (this paper)

- **May-analyses:**
  - **No** None
  - **RC** Based on relative competitiveness
  - **EMX** Early Miss eXploitation (this paper)

- **Instantiations of cache analysis framework:**
  - **No+CM**
  - **RC+CM**
  - **EMX+CM**

- **Synthetic benchmarks:**
  - Random access sequences and program fragments
Evaluation Results

- Average guaranteed hit- and miss-rates for a cache of size 8
- $n$ is number of pairwise different elements that are accessed
Outline

1 Introduction & Motivation
   - Timing Analysis
   - Cache Analysis

2 Abstract Interpretation of FIFO
   - Challenge FIFO Replacement
   - Domain Cooperation
   - Must Analysis
   - May Analysis

3 Evaluation
   - Related Work
   - Analysis Precision

4 Summary
Summary

- Cache analysis framework
  - Couple several analyses
  - Cooperation via classifications

- Canonical FIFO must-analysis
  - Potential misses

- EMX FIFO may-analysis
  - Definite misses
  - Early Miss eXploitation
Summary

- Cache analysis framework
  - Couple several analyses
  - Cooperation via classifications

- Canonical FIFO must-analysis
  - Potential misses

- EMX FIFO may-analysis
  - Definite misses
  - Early Miss eXploitation

\[ U_{\text{MUST}}([T_0, \ldots, T_{K-1}], t, cl) := \begin{cases} 
\emptyset, T_0, \ldots, T_{K-2} \cup \{t\} & : cl = \top \\
[T_0, \ldots, T_{K-1}] & : cl = H \\
[\{t\}, T_0, \ldots, T_{K-2}] & : cl = M 
\end{cases} \]

Thank you for listening. Questions?
Further Reading

R. Wilhelm et al.
The worst-case execution time problem— overview of methods and survey of tools
Transactions on Embedded Computing Systems, 7(3), 2008

J. Reineke and D. Grund
Relative competitive analysis of cache replacement policies
LCTES 2008
Related Work: LRU Analyses

- Bounds on number of cache misses:
  - Ghosh  Cache Miss Equations, loop nests
  - Chatterjee  Exact model of cache behavior for loop nests

- Classification of individual accesses:
  - Mueller  By “static cache simulation”
  - Li  By integer linear programming
  - Ferdinand  By abstract interpretation

- Only for LRU caches
What About The Gap for $n \leq k$?
Does the Initial Cache State Make a Difference?

- Yes, a FIFO of size \( k \), is \((k,k)\) sensitive

<table>
<thead>
<tr>
<th>Definition: Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy ( P ) is ((f, c)) miss-sensitive if</td>
</tr>
<tr>
<td>[ m_P(p, s) \leq f \cdot m_P(p', s) + c ]</td>
</tr>
<tr>
<td>for all access sequences ( s ) and all cache states ( p, p' ).</td>
</tr>
</tbody>
</table>

⇒ For FIFO of size 4, execution time may differ by a factor of 3

R. Wilhelm et al.
Memory Hierarchies, Pipelines, and Buses for Future Architectures in Time-critical Embedded Systems
*IEEE Transactions on CAD of Integrated Circuits and Systems 2009*