Precise and Efficient FIFO-Replacement Analysis Based on Static Phase Detection

Daniel Grund\textsuperscript{1} \quad Jan Reineke\textsuperscript{2}

\textsuperscript{1}Saarland University, Saarbrücken, Germany
\textsuperscript{2}University of California, Berkeley, USA

Euromicro Conference on Real-Time Systems 2010
Outline

1 Introduction and Problem
   - Timing Analysis
   - Cache Analysis
   - Challenge Replacement

2 Predicting Hits for
   - Idea and Theorem
   - Must Analysis
   - Efficient Implementation

3 Paper Contents

4 Evaluation
   - Related Work
   - Analysis Precision

5 Summary
Timing Analysis for Real-Time Systems

- Need to bound execution time of programs
- Execution time influenced by architectural features
  - pipelines, caches, branch prediction, ...
- Need to analyze behavior of architectural components
Caches and Replacement Policies

- Caches transparently buffer memory blocks

- Diagram showing CPU connected to Cache and then to Main Memory. The specifications are:
  - CPU
  - Cache: Capacity 32 KB, Latency 3 cycles
  - Main Memory: Capacity 2 MB, Latency 100 cycles
Caches and Replacement Policies

- Caches transparently buffer memory blocks
Caches and Replacement Policies

- Caches transparently buffer memory blocks

Diagram:
- CPU
- Cache
- Main Memory
- "hit" 
  (ab)
- Capacity: 32 KB
- Latency: 3 cycles
- 2 MB
- 100 cycles

Flow:
- CPU to Cache (a!)
- Cache to Main Memory

Diagram attributes:
- "hit": Red
- Flow: Red arrow
Caches and Replacement Policies

- Caches transparently buffer memory blocks
Caches and Replacement Policies

- Caches transparently buffer memory blocks

 laundry: 32 KB
Latency: 3 cycles

Main Memory
Capacity: 2 MB
Latency: 100 cycles
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

Diagram:

- CPU
- Cache
- Main Memory
  - Capacity: 32 KB
  - Latency: 3 cycles
  - 2 MB
  - 100 cycles

"miss" \((ac)\)
Caches and Replacement Policies

- Caches transparently buffer memory blocks
- Replacement policy \textit{dynamically} decides which element to replace
  - LRU least recently used
  - PLRU pseudo LRU
  - FIFO first-in first-out

![Diagram showing CPU, Cache, and Main Memory connections with cache miss (ac) and capacity, latency, and memory size details:]

- Capacity: 32 KB
- Latency: 3 cycles
- Main Memory: 2 MB
- 100 cycles
Static Cache Analysis
Goals & Notions

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

Must-information

- Underapproximation of cache contents
- Used to soundly classify cache hits

May-information

- Overapproximation of cache contents
- Used to soundly classify cache misses
Static Cache Analysis

Challenges
Static Cache Analysis

Challenges

Initial cache contents unknown
Static Cache Analysis

Challenges

- Initial cache contents unknown
- Need to combine analysis information
Static Cache Analysis

Challenges

- Initial cache contents unknown
- Need to combine analysis information
- Need to determine addresses of $x$, $y$, $z$
Static Cache Analysis

Challenges

Initial cache contents unknown
Need to combine analysis information
Need to determine addresses of $x, y, z$
FIFO Replacement

- FIFO cache of size $k$:

$$[b_1, \ldots, b_k] \in Q_k := \mathcal{B}_k$$

- Example updates:

$$[d, c, b, a] \xrightarrow{\text{hit}} [d, c, b, a]$$

$$[d, c, b, a] \xrightarrow{\text{miss}} [e, d, c, b]$$
FIFO Replacement Analysis

Why Predicting Hits is Difficult

- Take a set of blocks $B$ that does fit into a cache $q$
- For example, $B = \{a, b, e\}$ and $k = 4$. $|B| \leq k$.
- Access all blocks in $B$:
  $$q \xrightarrow{\langle a, b, e \rangle} q'$$

- Must all accessed blocks be cached? $\forall q : B \subseteq q'$?
FIFO Replacement Analysis

Why Predicting Hits is Difficult

- Take a set of blocks $B$ that does fit into a cache $q$
- For example, $B = \{a, b, e\}$ and $k = 4$. $|B| \leq k$.
- Access all blocks in $B$:

$$q \xrightarrow{\langle a, b, e \rangle} q'$$

- Must all accessed blocks be cached? $\forall q : B \subseteq q'$? No.

$$[d, c, b,a] \xrightarrow{a \text{ hit}} [d, c, b, a] \xrightarrow{b \text{ hit}} [d, c, b, a] \xrightarrow{e \text{ miss}} [e, d, c, b] \not\in a$$

Observation

After accessing a set of “fitting” blocks, not all of them must be cached.
FIFO Replacement Analysis

Why Predicting Misses is Difficult

- Take a set of blocks $B$ that does not fit into a cache $q$
- For example, $B = \{ a, b, c, d, e, f \}$ and $k = 4$. $|B| \geq k$.
- Access all blocks in $B$:

$$ q \xrightarrow{\langle a, b, c, d, e, f \rangle} q' $$

- Must all non-accessed blocks be evicted? $\forall q : q' \subseteq B$?
FIFO Replacement Analysis

Why Predicting Misses is Difficult

- Take a set of blocks $B$ that does not fit into a cache $q$
- For example, $B = \{a, b, c, d, e, f\}$ and $k = 4$. $|B| \geq k$.
- Access all blocks in $B$:

  $q \xrightarrow{\langle a,b,c,d,e,f\rangle} q'$

- Must all non-accessed blocks be evicted? $\forall q : q' \subseteq B$? No.

  $\exists c, b, a \xrightarrow{\langle a,b,c\rangle}_{\text{hits}} [x, c, b, a] \xrightarrow{\langle d,e,f\rangle}_{\text{misses}} [f, e, d, x]$ $\ni x$

Observation

After accessing a set of “non-fitting” blocks, other blocks may still be cached.
Outline

1 Introduction and Problem
   - Timing Analysis
   - Cache Analysis
   - Challenge Replacement

2 Predicting Hits for
   - Idea and Theorem
   - Must Analysis
   - Efficient Implementation

3 Paper Contents

4 Evaluation
   - Related Work
   - Analysis Precision

5 Summary
To the point: Anticipation & Idea

- Considering repeated accesses to “fitting” blocks $B$ helps:
  
  $B = \{a, b, c\}$
  
  $s = \langle a, b, b, c, b, b, a, c, c, a, b, \ldots \rangle$
  
  Eventually, all blocks in $B$ must be cached.

- Need to detect repetitions

- Partition access sequence $s$ into phases

**Definition (Phase)**

A $B$-phase is an access sequence $s$ such that the set of accessed blocks $A(s) = B$.

$$\langle a, b, b, c, b, b, a, c, \ldots \rangle$$

$\{a, b, c\}$-phase $\quad \{a, b, c\}$-phase
Predicting Hits by Detecting Phases

Lemma (Single Phase)

Let $s$ be a $B$-phase and $|B| \leq k$.

$$\forall q \in Q_k, q \xrightarrow{s} q' : \bigvee$$
Predicting Hits by Detecting Phases

Lemma (Single Phase)

Let $s$ be a $B$-phase and $|B| \leq k$.

$$\forall q \in Q_k, q \xrightarrow{s} q' : \ B \subseteq q' \lor$$

1. Either all blocks already cached:
   - $B \subseteq q \iff$ only hits in $s \Rightarrow B \subseteq q'$
Predicting Hits by Detecting Phases

Lemma (Single Phase)

Let $s$ be a $B$-phase and $|B| \leq k$.

$$\forall q \in Q_k, q \xrightarrow{s} q': B \subseteq q' \lor C_1(q') \subseteq B$$

1. Either all blocks already cached:
   - $B \subseteq q \Rightarrow$ only hits in $s \Rightarrow B \subseteq q'$

2. Or not:
   - $B \not\subseteq q \Rightarrow$ at least one miss $s \Rightarrow C_1(q') \subseteq B$
   - $[d, c, b, a] \xrightarrow{\langle a, b, e \rangle} [\underbrace{e}_{C_1(q') = \{e\} \subseteq B}, d, c, b]$
Theorem (Multiple Phases)

Let $s_i$ be $B$-phases and $|B| \leq k$ and $s = s_1 \circ \ldots \circ s_j$

$$\forall q \in Q_k, q \xrightarrow{s} q' : B \subseteq q' \lor C_j(q') \subseteq B$$

1. For each individual phase the lemma applies
2. Misses, if any, accumulate in last-in positions $C_j(q')$

$$[d, c, b, a] \xrightarrow{\langle a, b, e \rangle} [e, d, c, b] \xrightarrow{\langle b, a, e \rangle} [a, e, d, c] \xrightarrow{\langle a, b, e \rangle} [b, a, e, d]$$

$C_1 \subseteq B$

$C_2 \subseteq B$

$C_3 \subseteq B$
Predicting Hits by Detecting Phases

Theorem (Multiple Phases)

Let $s_i$ be $B$-phases and $|B| \leq k$ and $s = s_1 \circ \ldots \circ s_j$

$$\forall q \in Q_k, q \xrightarrow{s} q' : B \subseteq q' \lor C_j(q') \subseteq B$$

1. For each individual phase the lemma applies
2. Misses, if any, accumulate in last-in positions $C_j(q')$

$$[d, c, b, a] \xrightarrow{\langle a, b, e \rangle} [e, d, c, b] \xrightarrow{\langle b, a, e \rangle} [a, e, d, c] \xrightarrow{\langle a, b, e \rangle} [b, a, e, d]$$

- Corollary: After $|B|$ $B$-phases, all blocks in $B$ must be cached
The Must Analysis
How to Count Phases

- For phase blocks $B$, the analysis maintains:
  - $P$ phase progress, blocks already accessed in current phase
  - $pc$ phase counter, number of detected $B$-phases
- Predicts hits for blocks in $B$ if $pc = |B|$

Example for $B \equiv \{a, b\}$

<table>
<thead>
<tr>
<th>$s$</th>
<th>$a$</th>
<th>$b$</th>
<th>$b$</th>
<th>$b$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>$\emptyset$</td>
<td>${a}$</td>
<td>${a, b}$</td>
<td>$\emptyset$</td>
<td>${b}$</td>
<td>${b}$</td>
</tr>
<tr>
<td>$pc$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Hit</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$pc$ 2 Hit
The Must Analysis
Dependency on Future Accesses

- Need $|B|$ $B$-phases to predict hits for blocks in $B$
- How to choose $B$?
- After observing $\langle a, b, c \rangle$, it makes sense trying to detect
  - 2 further $\{a, b, c\}$-phases
  - 1 further $\{b, c\}$-phase
  - 0 further $\{c\}$-phases
- Optimal $B$ depends on future accesses
  - $\langle a, b, c, a, b, c, a, b, c \rangle$
  - $\langle a, b, c, b, c, b, c, b, c, b, c \rangle$
The Must Analysis
Resolving the Dependency

- Perform multiple analyses for different $B$ sets
- For which?

- $|B|$ already determines sensible contents of $B$
- For $|B| = 2$, after $\langle a, b, c \rangle$
  - already detected 1 $\{b, c\}$-phase
  - no advantage in trying to detect 2 $\{x, y\}$-phases

$\Rightarrow$ Perform $k$ analyses for different $B$ sets
  - for each phase size $n = 1 \ldots k$
  - $B_n$ consists of the $n$ most-recently-used blocks
The Must Analysis
Subanalyses for $n = 1 \ldots 3$

\[ a \quad b \quad c \quad c \quad b \quad c \quad a \quad a \quad c \quad a \quad b \quad a \]

$n = 1$:

$n = 2$:

$n = 3$:
The Must Analysis
Subanalyses for $n = 1 \ldots 3$

\[
\begin{align*}
& a \quad b \quad c \quad | \quad c \quad b \quad c \quad a \quad a \quad c \quad a \quad b \quad a \\
 n = 1 : & \quad \text{(c)} \\
 n = 2 : & \\
 n = 3 : &
\end{align*}
\]
The Must Analysis
Subanalis for $n = 1 \ldots 3$

\[
\begin{array}{cccc|cc|cccc|c}
\hline
a & b & c & c & b & c & a & c & a & b & a \\
\hline
n = 1 & & & & & & & & & & \\
\hline
n = 2 & & & & & & & & & & \\
\hline
n = 3 & & & & & & & & & & \\
\end{array}
\]

$H$

$H$

$\{c\}$

$\{b, c\}$

$\{b, c\}$
The Must Analysis
Subanalyses for \( n = 1 \ldots 3 \)

\[
\begin{array}{cccccccc}
  a & b & c & c & b & c & a & a & c & a & b & a \\
  H & H & & & & & & H & & & \\
\end{array}
\]

\( n = 1 : \)  
\{c\}  
\{a\}

\( n = 2 : \)  
\{b, c\}  
\{b, c\}

\( n = 3 : \)
The Must Analysis
Subanalyses for \( n = 1 \ldots 3 \)

\[
\begin{array}{cccccccc}
  a & b & c & c & b & c & a & a & c \\
  \text{H} & \text{H} & \text{H} & \text{H} & \text{H} & \text{H} \\
\end{array}
\]

\( n = 1 \) :
- \{c\} - \{a\} -

\( n = 2 \) :
- \{b, c\} - \{b, c\} - \{a, c\} - \{a, c\} -

\( n = 3 \) :
The Must Analysis
Subanalyses for $n = 1 \ldots 3$

\[
\begin{array}{cccccccc}
  & a & b & c & c & b & c & a & a & c & a & b & a \\
  & H & H & H & H & H & H & H & \\
 n = 1 : & \{c\} & \{a\} & \\
 n = 2 : & \{b, c\} & \{b, c\} & \{a, c\} & \{a, c\} & \\
 n = 3 : & \{a, b, c\} & \{a, b, c\} & \{a, b, c\} & \\
\end{array}
\]
Efficient Implementation

Observation

- For $n = 1 \ldots k$, analysis needs to maintain:
  - phase blocks $B_n \in 2^B$
  - phase progress $P_n \in 2^B$
  - phase counter $pc_n \in \mathbb{N}$

- Phase blocks $B_n$ are the $n$ most-recently-used blocks
  ⇒ For $i < j : B_i \subseteq B_j$
  ⇒ Encode all $B_n$ in a single LRU-stack

- For all $i : P_i \subseteq B_i$
  ⇒ Encode all $P_n$ as “pointers” into the stack
Efficient Implementation

Encoding

- For phase blocks $B_n$:
  - $pc_n$ complete $B_n$-phases were detected
  - current phase progress is $B_{pp_n}$

\[
\begin{array}{c|c}
B_1 & \{b\} \\
P_1 & \emptyset \\
pc_1 & 1 \\
B_2 & \{b, c\} \\
P_2 & \{b\} \\
pc_2 & 2 \\
B_3 & \{a, b, c\} \\
P_3 & \{b, c\} \\
pc_3 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c}
B_1 & pc_1, pp_1 \\
B_2 \setminus B_1 & pc_2, pp_2 \\
B_3 \setminus B_2 & pc_3, pp_3 \\
B_4 \setminus B_3 & pc_4, pp_4 \\
\end{array}
\]
Outline

1 Introduction and Problem
   - Timing Analysis
   - Cache Analysis
   - Challenge Replacement

2 Predicting Hits for
   - Idea and Theorem
   - Must Analysis
   - Efficient Implementation

3 Paper Contents

4 Evaluation
   - Related Work
   - Analysis Precision

5 Summary
Contents of the Paper

- So far, we have seen parts of the must-analysis
- The paper contains, for must- and may-analysis,
  - basic theorem
  - generalization to arbitrary control-flow
  - formalization as abstract interpretation
Outline

1 Introduction and Problem
   ▪ Timing Analysis
   ▪ Cache Analysis
   ▪ Challenge Replacement

2 Predicting Hits for
   ▪ Idea and Theorem
   ▪ Must Analysis
   ▪ Efficient Implementation

3 Paper Contents

4 Evaluation
   ▪ Related Work
   ▪ Analysis Precision

5 Summary
Brief History of Replacement Analysis

Before ’97  LRU analyses
LCTRTS’97 Precise and efficient must- and may-analysis for LRU [1]
LCTES’08 Generic analyses for FIFO and PLRU [2]
SAS’09 Cache analysis framework and FIFO analysis [3]
WCET’10 Toward precise analysis for PLRU [4]

ECRTS’10 Precise and efficient must- and may-analysis for FIFO
Evaluation Setup

- Analyses:
  - $C_m = HAM$ Must-analysis of SAS’09
  - RC Generic analyses of LCTES’08
  - PD Phase detecting analyses

- Collecting semantics:
  - CS Limit for any static analysis

- Spectrum of synthetic benchmarks:
  - Random access sequences and program fragments
  - With varying locality
Evaluation Results

$k=8$, random sequences

- $n$ is number of distinct elements that get accessed
- Average guaranteed hit- and miss-rates
Evaluation Results

$k=8$, random sequences

- $n$ is number of distinct elements that get accessed
- Average guaranteed hit- and miss-rates
Evaluation Results

$k=8$, random sequences

- $n$ is number of distinct elements that get accessed
- Average guaranteed hit- and miss-rates
Evaluation Results

$k=8$, random sequences

- $n$ is number of distinct elements that get accessed
- Average guaranteed hit- and miss-rates
Outline

1 Introduction and Problem
   - Timing Analysis
   - Cache Analysis
   - Challenge Replacement

2 Predicting Hits for
   - Idea and Theorem
   - Must Analysis
   - Efficient Implementation

3 Paper Contents

4 Evaluation
   - Related Work
   - Analysis Precision

5 Summary
Precise and Efficient
FIFO-Replacement Analysis based on Static Phase Detection
Summary

Precise and Efficient
FIFO-Replacement Analysis based on Static Phase Detection

- Two theorems on FIFO-contents
  - bound on number of phases
  - must be cached / evicted

- Must- and may-analysis
  - static phase detection
  - multiple sub-analyses
Further Reading

C. Ferdinand
Cache Behaviour Prediction for Real-Time Systems

J. Reineke and D. Grund
Relative competitive analysis of cache replacement policies
LCTES 2008

D. Grund and J. Reineke
Abstract Interpretation of FIFO Replacement
SAS 2009

D. Grund and J. Reineke
Toward Precise PLRU Cache Analysis
WCET 2010
Related Work: LRU Analyses

- Analyses directed at worst-case execution-time analysis
  - **Mueller**  By “static cache simulation”
  - **Li**       By integer linear programming
  - **Ferdinand** By abstract interpretation

- Other than that
  - **Ghosh**   Cache Miss Equations, loop nests
  - **Chatterjee** Exact model of cache behavior for loop nests

- All for LRU caches only
Static Timing-Analysis Framework

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
Applicability

- Any buffer with transparent FIFO replacement:
  - Individual cache sets of instruction of data caches (I\$, D\$)
  - Branch target buffers (BTB, BTIC)
  - Translation lookaside buffers (TLB)

- Instances:

  I\$ D\$
  - ARM 1136, 1156, 1176, 920T, 922T, 926EJ-S \((k \in \{2, 4, 64\})\)

  I\$ D\$
  - Marvell (Intel) XScale(s) \((k = 32)\)

  BTB
  - Freescale (Motorola) MPC 56x, 7450-Family \((k \in \{4, 8\})\)

  …
Must Analysis
Full Example for $k = 3$

- For $1 \leq n \leq k$ maintain $B_n, P_n, pc_n$

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
</tr>
<tr>
<td>$B_1$</td>
</tr>
<tr>
<td>$P_1$</td>
</tr>
<tr>
<td>$pc_1$</td>
</tr>
<tr>
<td>$B_2$</td>
</tr>
<tr>
<td>$P_2$</td>
</tr>
<tr>
<td>$pc_2$</td>
</tr>
<tr>
<td>$B_3$</td>
</tr>
<tr>
<td>$P_3$</td>
</tr>
<tr>
<td>$pc_3$</td>
</tr>
</tbody>
</table>

Hit

Hit
Must Analysis
Abstraction and Join

- Analysis domain is $Lru_k^{\leq} \times PC_k \times PP_k \subset (2^B)^k \times \mathbb{N}^k \times \mathbb{N}^k$
  - $Lru_k^{\leq}$: LRU must-analysis, under-approximates accessed blocks
  - $PC_k$: lower bounds on number of phases
  - $PP_k$: lower bounds on phase progress

- Reuse abstract transformer and join of $Lru_k^{\leq}$
- Define appropriately for $PC_k$ and $PP_k$
May-Analysis

- Similar to must-analysis
- Difference: Phases may be of different lengths and contents

Theorem (Multiple Phases)

\[ s = s_1 \circ \ldots \circ s_j, \forall i : |A(s_i)| = n_i \geq k: \]

\[ \forall q \in Q_k, q_s \rightarrow q' : C_{\sum_{i=1}^{j}(n_i-k+1)}(q') \subseteq A(s) = \bigcup_i A(s_i) \]

- More simultaneous sub-analyses
- Similar implementation employing LRU may-analysis