Caches in WCET Analysis
Predictability, Competitiveness, Sensitivity

Jan Reineke

November 7th, 2008
Outline

1. Introduction
   - WCET Analysis
   - Caches and Cache Analysis

2. Predictability Metrics

3. Relative Competitiveness

4. Sensitivity – Caches and Measurement-Based Timing Analysis

5. Summary
Outline

1. Introduction
   - WCET Analysis
   - Caches and Cache Analysis

2. Predictability Metrics

3. Relative Competitiveness

4. Sensitivity – Caches and Measurement-Based Timing Analysis

5. Summary
Controllers in planes, cars, plants, ... often have to satisfy hard real-time constraints

→ Need to statically derive upper bounds on WCETs of tasks

variation due to inputs and initial hardware state

BCET  ACET  WCET  upper bound  execution time
Caches

**How they work:**
- dynamically and transparently
- managed by replacement policy

![Diagram of CPU, Cache, and Main Memory with capacities and latencies]

- **Capacity:** 32 KB
- **Latency:** 3 cycles
- **Capacity:** 2 MB
- **Latency:** 100 cycles
Caches

How they work:
- dynamically and transparently
- managed by replacement policy

CPU ---[a?]--- Cache ---[“hit”][ab]--- Main Memory

Capacity:
- 32 KB
- 2 MB

Latency:
- 3 cycles
- 100 cycles
How they work:
- dynamically and transparently
- managed by replacement policy

CPU

Cache

Main Memory

Capacity: 32 KB
Latency: 3 cycles

Capacity: 2 MB
Latency: 100 cycles

"hit" [ab] → a!
Caches

How they work:
- dynamically and transparently
- managed by replacement policy

![Diagram showing CPU, Cache, and Main Memory with capacity and latency information]

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

“miss” [ab] → Cache
Caches

How they work:
- dynamically and transparently
- managed by replacement policy

CPU → Cache → Main Memory

Capacity:
- Cache: 32 KB
- Main Memory: 2 MB

Latency:
- Cache: 3 cycles
- Main Memory: 100 cycles

“miss” \([ab]\) → c?
Caches

How they work:

- dynamically and transparently
- managed by replacement policy

![Diagram of CPU, Cache, and Main Memory with cache hit and miss]

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

"miss" [ac]
Caches

- How they work:
  - dynamically and transparently
  - managed by replacement policy

![Diagram of CPU, Cache, and Main Memory with capacities and latencies]

- "miss" [ac]

- Cache analysis statically derives guarantees on cache behavior
Cache Analysis

Two types of cache analyses:

1. **Local guarantees: classification of individual accesses**
   - **May-Analysis** → Overapproximates cache contents
   - **Must-Analysis** → Underapproximates cache contents

2. **Global guarantees: bounds on cache hits/misses**
Cache Replacement Policies

- Least Recently Used (LRU) used in Intel Pentium I and MIPS 24K/34K
- First-In First-Out (FIFO or Round-Robin) used in Motorola PowerPC 56X, Intel XScale, ARM9, ARM11
- Pseudo-LRU (PLRU) used in Intel Pentium II-IV and PowerPC 75X
- Most Recently Used (MRU) as described in literature

Cache analyses almost exclusively for LRU
In practice: FIFO, PLRU, Pseudo Round-Robin, ...
Uncertainty in WCET Analysis

- Precision of WCET analysis determined by amount of uncertainty
- Uncertainty in cache analysis depends on replacement policy

![Diagram showing uncertainty in WCET analysis](image)

- BCET
- ACET
- WCET upper bound
- execution time

uncertainty × penalty

variation due to inputs and initial hardware state
Outline

1. Introduction
   - WCET Analysis
   - Caches and Cache Analysis

2. Predictability Metrics

3. Relative Competitiveness

4. Sensitivity – Caches and Measurement-Based Timing Analysis

5. Summary
1. Initial cache contents unknown.
2. Need to combine information.
3. Cannot resolve address of $z$. Amount of uncertainty determined by ability to recover information.
1. Initial cache contents unknown.

- read z
- read y
- read x
- write z
Uncertainty in Cache Analysis

1. Initial cache contents unknown.
2. Need to combine information.
Uncertainty in Cache Analysis

1. Initial cache contents unknown.
2. Need to combine information.
3. Cannot resolve address of $z$.

Amount of uncertainty determined by ability to recover information.
Uncertainty in Cache Analysis

1. Initial cache contents unknown.
2. Need to combine information.
3. Cannot resolve address of z.

Amount of uncertainty determined by ability to recover information
Sequence: \( \langle a, \ldots, e, f, g, h \rangle \)
Meaning of Metrics

- **Evict**
  - Number of accesses to obtain *any* *may*-information.
  - I.e. when can an analysis predict any cache misses?

- **Fill**
  - Number of accesses to complete *may*- and *must*-information.
  - I.e. when can an analysis predict each access?

→ Evict and Fill bound the precision of *any* static cache analysis. Can thus serve as a benchmark for analyses.
### Evaluation of Policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Evict($k$)</th>
<th>Fill($k$)</th>
<th>Evict(8)</th>
<th>Fill(8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>$k$</td>
<td>$k$</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>FIFO</td>
<td>$2k - 1$</td>
<td>$3k - 1$</td>
<td>15</td>
<td>23</td>
</tr>
<tr>
<td>MRU</td>
<td>$2k - 2$</td>
<td>$\infty / 3k - 4$</td>
<td>14</td>
<td>$\infty / 20$</td>
</tr>
<tr>
<td>PLRU</td>
<td>$\frac{k}{2} \log_2 k + 1$</td>
<td>$\frac{k}{2} \log_2 k + k - 1$</td>
<td>13</td>
<td>19</td>
</tr>
</tbody>
</table>

- LRU is optimal w.r.t. metrics.
- Other policies are much less predictable.
- Use LRU.

How to obtain *may*- and *must*-information within the given limits for other policies?
Outline

1. Introduction
   - WCET Analysis
   - Caches and Cache Analysis

2. Predictability Metrics

3. Relative Competitiveness

4. Sensitivity – Caches and Measurement-Based Timing Analysis

5. Summary
Relative Competitiveness

- **Competitiveness** (Sleator and Tarjan, 1985): worst-case performance of an online policy *relative to the optimal offline policy*
  - used to evaluate online policies

- **Relative competitiveness** (Reineke and Grund, 2008): worst-case performance of an online policy *relative to another online policy*
  - used to derive local and global cache analyses
**Definition – Relative Miss-Competitiveness**

**Notation**

\[
m_p(p, s) = \text{number of misses that policy } P \text{ incurs on access sequence } s \in M^* \text{ starting in state } p \in C^P
\]
Definition – Relative Miss-Competitiveness

Notation

\[ m_P(p, s) = \text{number of misses that policy } P \text{ incurs on access sequence } s \in M^* \text{ starting in state } p \in C^P \]

Definition (Relative miss competitiveness)

Policy \( P \) is \((k, c)\)-miss-competitive relative to policy \( Q \) if

\[ m_P(p, s) \leq k \cdot m_Q(q, s) + c \]

for all access sequences \( s \in M^* \) and cache-set states \( p \in C^P, q \in C^Q \) that are compatible \( p \sim q \).
Definition – Relative Miss-Competitiveness

Notation

\[ m_P(p, s) = \text{number of misses that policy } P \text{ incurs on access sequence } s \in M^* \text{ starting in state } p \in C_P \]

Definition (Relative miss competitiveness)

Policy \( P \) is \((k, c)\)-miss-competitive relative to policy \( Q \) if

\[ m_P(p, s) \leq k \cdot m_Q(q, s) + c \]

for all access sequences \( s \in M^* \) and cache-set states \( p \in C_P, q \in C_Q \) that are compatible \( p \sim q \).

Definition (Competitive miss ratio of \( P \) relative to \( Q \))

The smallest \( k \), s.t. \( P \) is \((k, c)\)-miss-competitive rel. to \( Q \) for some \( c \).
Example – Relative Miss-Competitiveness

\( P \) is \((3, 4)\)-miss-competitive relative to \( Q \).
If \( Q \) incurs \( x \) misses, then \( P \) incurs at most \( 3 \cdot x + 4 \) misses.
Example – Relative Miss-Competitiveness

\( \mathbf{P} \) is \((3, 4)\)-miss-competitive relative to \( \mathbf{Q} \).
If \( \mathbf{Q} \) incurs \( x \) misses, then \( \mathbf{P} \) incurs at most \( 3 \cdot x + 4 \) misses.

**Best:** \( \mathbf{P} \) is \((1, 0)\)-miss-competitive relative to \( \mathbf{Q} \).
Example – Relative Miss-Competitiveness

\( P \) is \((3, 4)\)-miss-competitive relative to \( Q \).
If \( Q \) incurs \( x \) misses, then \( P \) incurs at most \( 3 \cdot x + 4 \) misses.

Best: \( P \) is \((1, 0)\)-miss-competitive relative to \( Q \).

Worst: \( P \) is not-miss-competitive (or \( \infty \)-miss-competitive) relative to \( Q \).
Example – Relative Hit-Competitiveness

**P** is \(\left(\frac{2}{3}, 3\right)\)-hit-competitive relative to **Q**.
If **Q** has \(x\) hits, then **P** has at least \(\frac{2}{3} \cdot x - 3\) hits.
Example – Relative Hit-Competitiveness

\( P \) is \( \left( \frac{2}{3}, 3 \right) \)-hit-competitive relative to \( Q \).

If \( Q \) has \( x \) hits, then \( P \) has at least \( \frac{2}{3} \cdot x - 3 \) hits.

**Best:** \( P \) is \( (1, 0) \)-hit-competitive relative to \( Q \).
Equivalent to \( (1, 0) \)-miss-competitiveness.

**Worst:** \( P \) is \( (0, 0) \)-hit-competitive relative to \( Q \).
Analogue to \( \infty \)-miss-competitiveness.
Example – Relative Hit-Competitiveness

\( \mathbf{P} \) is \( \left( \frac{2}{3}, 3 \right) \)-hit-competitive relative to \( \mathbf{Q} \).
If \( \mathbf{Q} \) has \( x \) hits, then \( \mathbf{P} \) has at least \( \frac{2}{3} \cdot x - 3 \) hits.

**Best:** \( \mathbf{P} \) is \( (1, 0) \)-hit-competitive relative to \( \mathbf{Q} \).
Equivalent to \( (1, 0) \)-miss-competitiveness.

**Worst:** \( \mathbf{P} \) is \( (0, 0) \)-hit-competitive relative to \( \mathbf{Q} \).
Analogue to \( \infty \)-miss-competitiveness.
Local Guarantees: (1, 0)-Competitiveness

Let $P$ be $(1, 0)$-competitive relative to $Q$:

$$m_P(p, s) \leq 1 \cdot m_Q(q, s) + 0$$

$$\Leftrightarrow m_P(p, s) \leq m_Q(q, s)$$
Local Guarantees: (1, 0)-Competitiveness

Let $P$ be $(1, 0)$-competitive relative to $Q$:

$$m_P(p, s) \leq 1 \cdot m_Q(q, s) + 0$$

$$\Leftrightarrow m_P(p, s) \leq m_Q(q, s)$$

1. If $Q$ “hits”, so does $P$, and
2. if $P$ “misses”, so does $Q$. 

Local Guarantees: (1, 0)-Competitiveness

Let $P$ be (1, 0)-competitive relative to $Q$:

$$m_P(p, s) \leq 1 \cdot m_Q(q, s) + 0$$

$$\Leftrightarrow m_P(p, s) \leq m_Q(q, s)$$

1. If $Q$ “hits”, so does $P$, and
2. if $P$ “misses”, so does $Q$.

As a consequence,

1. a must-analysis for $Q$ is also a must-analysis for $P$, and
2. a may-analysis for $P$ is also a may-analysis for $Q$. 
Global Guarantees: \((k, c)\)-Competitiveness

Given: Global guarantees for policy \(Q\).
Wanted: Global guarantees for policy \(P\).
Global Guarantees: \((k, c)\)-Competitiveness

**Given:** Global guarantees for policy \(Q\).

**Wanted:** Global guarantees for policy \(P\).

1. Determine competitiveness of policy \(P\) relative to policy \(Q\).

\[
m_P \leq k \cdot m_Q + c
\]
Global Guarantees: \((k, c)\)-Competitiveness

**Given:** Global guarantees for policy \(Q\).

**Wanted:** Global guarantees for policy \(P\).

1. Determine competitiveness of policy \(P\) relative to policy \(Q\).
   \[
   m_P \leq k \cdot m_Q + c
   \]

2. Compute global guarantee for task \(T\) under policy \(Q\).
   \[
   m_Q(T)
   \]
Global Guarantees: \((k, c)\)-Competitiveness

Given: Global guarantees for policy \(Q\).
Wanted: Global guarantees for policy \(P\).

1. Determine competitiveness of policy \(P\) relative to policy \(Q\).
   \[ m_P \leq k \cdot m_Q + c \]

2. Compute global guarantee for task \(T\) under policy \(Q\).
   \[ m_Q(T) \]

3. Calculate global guarantee on the number of misses for \(P\) using the global guarantee for \(Q\) and the competitiveness results of \(P\) relative to \(Q\).
   \[ m_P \leq k \cdot m_Q + c = m_P(T) \]
Relative Competitiveness – Automatic Computation

**P** and **Q** (here: FIFO and LRU) induce transition system:

\[
\begin{align*}
&\text{(h, h)} \quad \xrightarrow{e} \\
&[eabc]_{\text{FIFO}}, [eabc]_{\text{LRU}} \\
&\text{c} \quad (h, h) \quad \downarrow \\
&[eabc]_{\text{FIFO}}, [ceab]_{\text{LRU}} \\
&\text{e} \quad (m, m) \quad \xrightarrow{e} \\
&[eabc]_{\text{FIFO}}, [eabc]_{\text{LRU}} \\
&\text{f} \quad \text{first-in} \\
&\text{MRU} \\
&\text{LRU} \\
&\text{Legend} \\
&[abcd]_{\text{FIFO}}, [abcd]_{\text{LRU}} \\
&\text{d} \quad (h, h) \quad \downarrow \\
&[abcd]_{\text{FIFO}}, [dabc]_{\text{LRU}} \\
&\text{e} \quad (m, m) \quad \xrightarrow{e} \\
&[eabc]_{\text{FIFO}}, [ceda]_{\text{LRU}} \\
&\text{c} \quad (h, m) \quad \downarrow \\
&[eabc]_{\text{FIFO}}, [ceda]_{\text{LRU}} \\
&\text{d} \quad (h, h) \quad \downarrow \\
&[deab]_{\text{FIFO}}, [deab]_{\text{LRU}} \\
&\text{Legend} \\
&[abcd]_{\text{FIFO}}, [abcd]_{\text{LRU}} \\
&\text{c} \quad (h, m) \quad \downarrow \\
&[eabc]_{\text{FIFO}}, [edab]_{\text{LRU}} \\
&\text{d} \quad (m, h) \quad \xrightarrow{d} \\
&[deab]_{\text{FIFO}}, [deab]_{\text{LRU}} \\
\end{align*}
\]

**Competitive miss ratio** = maximum ratio of misses in policy **P** to misses in policy **Q** in transition system
Problem: The induced transition system is $\infty$ large.
Observation: Only the relative positions of elements matter:

\[
\begin{aligned}
[abc]_{\text{LRU}}, [bde]_{\text{FIFO}} & \approx [fgl]_{\text{LRU}}, [ghm]_{\text{FIFO}} \\
\quad c \ (h, m) & \quad l \ (h, m) \\
[cab]_{\text{LRU}}, [cbd]_{\text{FIFO}} & \approx [lfg]_{\text{LRU}}, [lgh]_{\text{FIFO}}
\end{aligned}
\]

Solution: Construct finite quotient transition system.
\( \approx \)-Equivalent States in Running Example

\[
\begin{align*}
\{eabc\}_{\text{FIFO}}, \{eabc\}_{\text{LRU}} & \xrightarrow{e} \{abcd\}_{\text{FIFO}}, \{abcd\}_{\text{LRU}} \\
\{eabc\}_{\text{FIFO}}, \{ceab\}_{\text{LRU}} & \xrightarrow{d} \{abcd\}_{\text{FIFO}}, \{dabc\}_{\text{LRU}} \\
\{eabc\}_{\text{FIFO}}, \{ceda\}_{\text{LRU}} & \xrightarrow{d} \{deab\}_{\text{FIFO}}, \{deab\}_{\text{LRU}} \\
\{eabc\}_{\text{FIFO}}, \{edab\}_{\text{LRU}} & \xrightarrow{d} \{deab\}_{\text{FIFO}}, \{deab\}_{\text{LRU}}
\end{align*}
\]
Merging \(\approx\)-equivalent states yields a finite quotient transition system:
Competitive Ratio = Maximum Cycle Ratio

Competitive miss ratio =

maximum ratio of misses in policy $P$ to misses in policy $Q$
Competitive Ratio = Maximum Cycle Ratio

Competitive miss ratio =
maximum ratio of misses in policy \( P \) to misses in policy \( Q \)

Maximum cycle ratio = \( \frac{0 + 1 + 1}{0 + 1 + 0} = 2 \)
Tool Implementation

- Implemented in Java
- Interface for replacement policies
- Fully automatic
- Provides example sequences for competitive ratio and constant

- Analysis usually practically feasible up to associativity 8
  - limited by memory consumption
  - depends on similarity of replacement policies
Generalizations

Identified patterns and proved generalizations by hand. Aided by example sequences generated by tool.
Generalizations

Identified patterns and proved generalizations by hand. Aided by example sequences generated by tool.

Previously unknown facts:

\[ \text{PLRU}(k) \text{ is } (1,0) \text{ comp. rel. to } \text{LRU}(1 + \log_2 k), \]

\[ \rightarrow \text{LRU-}must\text{-analysis can be used for PLRU} \]
Generalizations

Identified patterns and proved generalizations by hand. Aided by example sequences generated by tool.

Previously unknown facts:

\[ \text{PLRU}(k) \text{ is } (1, 0) \text{ comp. rel. to } \text{LRU}(1 + \log_2 k), \]
\[ \rightarrow \text{LRU-\textit{must}-analysis can be used for PLRU} \]

\[ \text{FIFO}(k) \text{ is } \left(\frac{1}{2}, \frac{k-1}{2}\right) \text{ hit-comp. rel. to } \text{LRU}(k), \text{ whereas} \]

\[ \text{LRU}(k) \text{ is } \textit{not} \text{ hit-comp. rel. to } \text{FIFO}(k), \text{ but} \]
Generalizations

Identified patterns and proved generalizations by hand. Aided by example sequences generated by tool.

Previously unknown facts:

- $\text{PLRU}(k)$ is $(1, 0)$ comp. rel. to $\text{LRU}(1 + \log_2 k)$, $\longrightarrow$ LRU-\textit{must}-analysis can be used for PLRU

- $\text{FIFO}(k)$ is $(\frac{1}{2}, \frac{k-1}{2})$ hit-comp. rel. to $\text{LRU}(k)$, whereas
- $\text{LRU}(k)$ is \textit{not} hit-comp. rel. to $\text{FIFO}(k)$, but

- $\text{LRU}(2k - 1)$ is $(1, 0)$ comp. rel. to $\text{FIFO}(k)$, and
- $\text{LRU}(2k - 2)$ is $(1, 0)$ comp. rel. to $\text{MRU}(k)$.
   $\longrightarrow$ LRU-\textit{may}-analysis can be used for FIFO and MRU
   $\longrightarrow$ optimal with respect to predictability metrics
Generalizations

Identified patterns and proved generalizations by hand. Aided by example sequences generated by tool.

Previously unknown facts:

\[
\text{PLRU}(k) \text{ is } (1, 0) \text{ comp. rel. to } \text{LRU}(1 + \log_2 k),
\]

\[\rightarrow \text{LRU}-\text{must-analysis can be used for PLRU}\]

\[
\text{FIFO}(k) \text{ is } \left(\frac{1}{2}, \frac{k-1}{2}\right) \text{ hit-comp. rel. to LRU}(k), \text{ whereas}
\]

\[
\text{LRU}(k) \text{ is } \text{not} \text{ hit-comp. rel. to FIFO}(k), \text{ but}
\]

\[
\text{LRU}(2k - 1) \text{ is } (1, 0) \text{ comp. rel. to FIFO}(k), \text{ and}
\]

\[
\text{LRU}(2k - 2) \text{ is } (1, 0) \text{ comp. rel. to MRU}(k).
\]

\[\rightarrow \text{LRU}-\text{may-analysis can be used for FIFO and MRU}\]

\[\rightarrow \text{optimal with respect to predictability metrics}\]

\[\text{FIFO}-\text{may-analysis used in the analysis of the branch target buffer of the MOTOROLA POWERPC 56X.}\]
Measurement-Based Timing Analysis

- Run program on a number of inputs and initial states.
- Combine measurements for basic blocks to obtain WCET estimation.
- Sensitivity Analysis demonstrates this approach may be dramatically wrong.
Run program on a number of inputs and initial states.

Combine measurements for basic blocks to obtain WCET estimation.

Sensitivity Analysis demonstrates this approach may be dramatically wrong.
Influence of Initial Cache State

variation due to initial cache state

BCET  \[ \rightarrow \]  WCET  \[ \leftarrow \]  upper bound  \[ \rightarrow \]  execution time

Definition (Miss sensitivity)

Policy $P$ is $(k, c)$-miss-sensitive if

$$m_P(p, s) \leq k \cdot m_P(p', s) + c$$

for all access sequences $s \in M^*$ and cache-set states $p, p' \in C^P$. 
### Sensitivity Results

<table>
<thead>
<tr>
<th>Policy</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>1,2</td>
<td>1,3</td>
<td>1,4</td>
<td>1,5</td>
<td>1,6</td>
<td>1,7</td>
<td>1,8</td>
</tr>
<tr>
<td>FIFO</td>
<td>2,2</td>
<td>3,3</td>
<td>4,4</td>
<td>5,5</td>
<td>6,6</td>
<td>7,7</td>
<td>8,8</td>
</tr>
<tr>
<td>PLRU</td>
<td>1,2</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>MRU</td>
<td>1,2</td>
<td>3,4</td>
<td>5,6</td>
<td>7,8</td>
<td>MEM</td>
<td>MEM</td>
<td>MEM</td>
</tr>
</tbody>
</table>

- LRU is optimal. Performance varies in the least possible way.
- For FIFO, PLRU, and MRU the number of misses may vary strongly.
- Case study based on simple model of execution time by Hennessy and Patterson (2003): WCET may be 3 times higher than a measured execution time for 4-way FIFO.
Outline

1. Introduction
   - WCET Analysis
   - Caches and Cache Analysis

2. Predictability Metrics

3. Relative Competitiveness

4. Sensitivity – Caches and Measurement-Based Timing Analysis

5. Summary
Summary

Predictability Metrics

...bound the precision of \textit{any} static cache analysis,
...quantify the predictability of replacement policies.
\rightarrow \text{LRU is the most predictable policy.}
Summary

Predictability Metrics

- ... bound the precision of *any* static cache analysis,
- ... quantify the predictability of replacement policies.
- → LRU is the most predictable policy.

Relative Competitiveness

- ... allows to derive guarantees on cache performance,
- ... can be computed automatically by building quotient system,
- ... yields first *may*-analyses for FIFO and MRU.

Thank you for your attention!
Summary

Predictability Metrics

... bound the precision of any static cache analysis,
... quantify the predictability of replacement policies.

→ LRU is the most predictable policy.

Relative Competitiveness

... allows to derive guarantees on cache performance,
... can be computed automatically by building quotient system,
... yields first may-analyses for FIFO and MRU.

Sensitivity Analysis

... determines the influence of initial state on cache performance,
... shows that measurement-based WCET analysis may be dramatically wrong.
Summary
Predictability Metrics
... bound the precision of any static cache analysis,
... quantify the predictability of replacement policies.
→ LRU is the most predictable policy.

Relative Competitiveness
... allows to derive guarantees on cache performance,
... can be computed automatically by building quotient system,
... yields first may-analyses for FIFO and MRU.

Sensitivity Analysis
... determines the influence of initial state on cache performance,
... shows that measurement-based WCET analysis may be dramatically wrong.

Thank you for your attention!
Predictability Metrics

...bound the precision of any static cache analysis,
...quantify the predictability of replacement policies.
→ LRU is the most predictable policy.

Relative Competitiveness

...allows to derive guarantees on cache performance,
...can be computed automatically by building quotient system,
...yields first may-analyses for FIFO and MRU.

Sensitivity Analysis

...determines the influence of initial state on cache performance,
...shows that measurement-based WCET analysis may be dramatically wrong.

Thank you for your attention!
MRU-bits record whether line was recently used

- **[abcd]_0101** → b,d
- **[ebcd]_1101** → e,b,d
- **[ebcd]_0010** → c

→ Never converges
Pseudo-LRU – PLRU

Initial cache-set state \([a, b, c, d]_{110}\).

After a miss on \(e\). State: \([a, b, e, d]_{011}\).

After a hit on \(a\). State: \([a, b, e, d]_{111}\).

After a miss on \(f\). State: \([a, b, e, f]_{010}\).

Hit on \(a\) “rejuvenates” neighborhood; “saves” \(b\) from eviction.
May- and Must-Information

\[
May^P(s) := \bigcup_{p \in C^P} CC^P(update^P(p, s))
\]

\[
Must^P(s) := \bigcap_{p \in C^P} CC^P(update^P(p, s))
\]

\[
may^P(n) := \left|May^P(s)\right|, \text{ where } s \in S^\neq \subsetneq M^*, |s| = n
\]

\[
must^P(n) := \left|Must^P(s)\right|, \text{ where } s \in S^\neq \subsetneq M^*, |s| = n
\]

\(S^\neq\) : set of finite access sequences with pairwise different accesses
Definitions of Metrics

$$\text{Evict}^P := \min \left\{ n \mid \text{may}^P(n) \leq n \right\},$$

$$\text{Fill}^P := \min \left\{ n \mid \text{must}^P(n) = k \right\},$$

where $k$ is $P$'s associativity.
Relation: Pred. Metrics ↔ Rel. Competitiveness

Let $P(k)$ be $(1, 0)$-miss-competitive relative to policy $Q(l)$, then

(i) $Evict^P(k) \geq Evict^Q(l)$,

(ii) $mls^P(k) \geq mls^Q(l)$. 
Let \( l \) be the smallest associativity, such that LRU\((l)\) is \((1, 0)\)-miss-competitive relative to \( P(k) \). Then

\[
\text{Alt-Evict}^P(k) = l.
\]

Let \( l \) be the greatest associativity, such that \( P(k) \) is \((1, 0)\)-miss-competitive relative to LRU\((l)\). Then

\[
\text{Alt-mls}^P(k) = l.
\]
Size of Transition System

\[2^{l+l'} \cdot \sum_{i=0}^{k} \binom{k}{i} \cdot \sum_{i'=0}^{k'} \binom{k'}{i'} \cdot \sum_{j=0}^{\min\{i,i'\}} \left(\binom{j}{i}\binom{j}{i'}\right) j!\]

\[\text{status bits of } \mathbf{P} \text{ and } \mathbf{Q} \]
\[\text{non-empty lines in } \mathbf{P} \]
\[\text{non-empty lines in } \mathbf{Q} \]
\[\text{number of overlappings in non-empty lines}\]

\[\min\{k,k'\} \sum_{j=0}^{\infty} \binom{k}{j} \binom{k'}{j} j! \leq k! \cdot k'! \sum_{j=0}^{\infty} \frac{1}{(k-j)!j!(k'-j)!}\]

\[\leq k! \cdot k'! \sum_{j=0}^{\infty} \frac{1}{j!} = e \cdot k! \cdot k'!\]

This can be bounded by

\[2^{l+l'+k+k'} \leq \|((C_k^l \times C_k^{l'})^*)\| \approx | \leq 2^{l+l'+k+k'} \cdot e \cdot k! \cdot k'!\]

bound on number of overlappings
Set-Associative Caches

Address:

\[ \log_2(s) \quad \log_2(8 \times b) \]

<table>
<thead>
<tr>
<th>Tag</th>
<th>Index</th>
<th>Block offset</th>
</tr>
</thead>
</table>

Cache Set:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Data Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Data Block</td>
</tr>
<tr>
<td>Tag</td>
<td>Data Block</td>
</tr>
</tbody>
</table>

Cache Set:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Data Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Data Block</td>
</tr>
<tr>
<td>Tag</td>
<td>Data Block</td>
</tr>
</tbody>
</table>

MUX

Data

\[ S \]

\[ k \]

Yes: Hit!

No: Miss!
Compatible States

\[ i^P = [\bot \bot \bot \bot]_P \approx i^Q = [\bot \bot \bot \bot]_Q \]

\[ update_P(i^P, s) \approx update_Q(i^Q, s) \]
Let $P$ be $(1, 0)$-competitive relative to $Q$, then

$$m_P(p, \langle x \rangle) = 1 \quad \implies \quad m_Q(q, \langle x \rangle) = 1$$
(1, 0)-Competitiveness and May/Must-Analyses

\[ \forall p \in P : m_P(p, \langle x \rangle) = 1 \]

\[ \Rightarrow \forall q \in Q : m_Q(q, \langle x \rangle) = 1 \]
Case Study: Impact of Sensitivity

- Simple model of execution time from Hennessy & Patterson (2003)
- $\text{CPI}_{hit} = \text{Cycles per instruction assuming cache hits only}$
- $\frac{\text{Memory accesses}}{\text{Instruction}}$ including instruction and data fetches

\[
\frac{T_{wc}}{T_{meas}} = \frac{\text{CPI}_{hit} + \frac{\text{Memory accesses}}{\text{Instruction}} \times \text{Miss rate}_{wc} \times \text{Miss penalty}}{\text{CPI}_{hit} + \frac{\text{Memory accesses}}{\text{Instruction}} \times \text{Miss rate}_{meas} \times \text{Miss penalty}}
\]

\[
= \frac{1.5 + 1.2 \times 0.20 \times 50}{1.5 + 1.2 \times 0.05 \times 50} = \frac{13.5}{4.5} = 3
\]