Design and Analysis of Real-Time Systems
Caches in WCET Analysis

Jan Reineke
Department of Computer Science
Saarland University
Saarbrücken, Germany

Advanced Lecture, Summer 2013
Outline

1. Caches
2. Cache Analysis for Least-Recently-Used
3. Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis
4. Summary
Outline

1 Caches

2 Cache Analysis for Least-Recently-Used

3 Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis

4 Summary
Caches

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

Why they work: *principle of locality*
- spatial
- temporal

[Diagram of CPU, Cache, Main Memory with capacities and latencies: 32 KB, 3 cycles for the cache; 2 MB, 100 cycles for main memory]
Caches

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

![Diagram of CPU, Cache, and Main Memory]

- Why they work: *principle of locality*
  - spatial
  - temporal

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

- Latency:
  - CPU: 3 cycles
  - Main Memory: 100 cycles
Caches

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

- Why they work: *principle of locality*
  - spatial
  - temporal

[Diagram showing CPU, Cache, Main Memory connections and cache capacity and latency values: 32 KB/3 cycles, 2 MB/100 cycles]
Caches

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

- Why they work: *principle of locality*
  - spatial
  - temporal

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

- Capacities:
  - CPU Cache: 32 KB
  - Main Memory: 2 MB

- Latencies:
  - CPU Cache: 3 cycles
  - Main Memory: 100 cycles

“miss” [ab]
Caches

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

- Why they work: *principle of locality*
  - spatial
  - temporal

![Diagram of cache system]

- **CPU**
  - Capacity: 32 KB
  - Latency: 3 cycles

- **Cache**
  - "miss" [ab]

- **Main Memory**
  - 2 MB
  - 100 cycles
Caches

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

![Diagram of CPU, Cache, and Main Memory with "miss" and formulas]

- Why they work: *principle of locality*
  - spatial
  - temporal

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

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

\[ c_3 = \langle c_1 c_2 c_3 c_4 \rangle ! \]
Caches

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

```
CPU                      Cache                      Main Memory
```

```
Capacity: 32 KB
Latency: 3 cycles
```

```
Capacity: 2 MB
Latency: 100 cycles
```

- Why they work: *principle of locality*
  - spatial
  - temporal
Caches

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

- Why they work: *principle of locality*
  - spatial
  - temporal

```
CPU -> Cache ("hit") [ac] -> Main Memory
```

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

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

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

```
<table>
<thead>
<tr>
<th>CPU</th>
<th>Cache</th>
<th>Main Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity: 32 KB</td>
<td></td>
<td>2 MB</td>
</tr>
<tr>
<td>Latency: 3 cycles</td>
<td></td>
<td>100 cycles</td>
</tr>
</tbody>
</table>
```

Why they work: *principle of locality*
- spatial
- temporal
Fully-Associative Caches

Address:

Tag

Block offset

$\log_2(8 \times b)$

MUX

Data

Yes: Hit!

No: Miss!

$k = \text{associativity}$

Table:

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

$=?$
Set-Associative Caches

Special cases:
- direct-mapped cache: only one line per cache set
- fully-associative cache: only one cache set
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

Each cache set is treated independently:

→ Set-associative caches are compositions of fully-associative caches.
Outline

1. Caches

2. Cache Analysis for Least-Recently-Used

3. Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis

4. Summary
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 analyses almost exclusively for LRU
- In practice: FIFO, PLRU, ...
Challenges for Cache Analysis

Always a cache hit/always a miss?

1. Initial cache contents unknown.
2. Different paths lead to these points.
3. Cannot resolve address of $z$.

Jan Reineke
Caches in WCET Analysis
Advanced Lecture, 2013
Challenges for Cache Analysis

Always a cache hit/always a miss?

1. Initial cache contents unknown.
2. Different paths lead to these points.
3. Cannot resolve address of $z$. 

Jan Reineke
Caches in WCET Analysis
Advanced Lecture, 2013
Deriving Invariants about Cache States using Abstract Interpretation

Collecting Semantics = set of states at each program point that any execution may encounter there

Two approximations:

- Collecting Semantics uncomputable
- Cache Semantics computable
- $\subseteq (Abstract$ $Cache$ $Sem.)$ efficiently computable
Deriving Invariants about Cache States using Abstract Interpretation

Collecting Semantics = set of states at each program point that any execution may encounter there

Two approximations:

- Collecting Semantics uncomputable
- \( \subseteq \) Cache Semantics computable
- \( \subseteq \gamma(A) \) (Abstract Cache Sem.) efficiently computable
Deriving Invariants about Cache States using Abstract Interpretation

Collecting Semantics = set of states at each program point that any execution may encounter there

Two approximations:

- Collecting Semantics uncomputable
- \( \subseteq \) Cache Semantics computable
- \( \subseteq \gamma(\text{Abstract Cache Sem.}) \) efficiently computable
Deriving Invariants about Cache States using Abstract Interpretation

Collecting Semantics = set of states at each program point that any execution may encounter there

Two approximations:
- Collecting Semantics uncomputable
- Cache Semantics computable
- $\subseteq$ Actual Cache Semantics
- $\subseteq$ $\gamma$(Abstract Cache Sem.) efficiently computable
Deriving Invariants about Cache States using Abstract Interpretation

**Collecting Semantics** =
set of states at each program point that any execution may encounter there

Two approximations:
- Collecting Semantics uncomputable
- $\subseteq$ Cache Semantics computable
- $\subseteq \gamma$(Abstract Cache Sem.) efficiently computable
Least-Recently-Used (LRU): Concrete Behavior

“Cache Miss”:

```
  z
  y
  x
  t
```

LRU has notion of age

```
  s
  z
  y
  x
```

“Cache Hit”:

```
  z
  y
  s
  t
```

```
  s
  z
  y
  t
```
LRU: Must-Analysis: Abstract Domain

- Used to predict *cache hits*.
- Maintains *upper bounds on ages* of memory blocks.
- Upper bound $\leq$ associativity $\rightarrow$ memory block definitely cached.

**Example**

<table>
<thead>
<tr>
<th>Abstract state:</th>
<th>age 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>${x}$</td>
<td></td>
</tr>
<tr>
<td>${}$</td>
<td></td>
</tr>
<tr>
<td>${s,t}$</td>
<td></td>
</tr>
<tr>
<td>${}$</td>
<td></td>
</tr>
</tbody>
</table>

\[ \gamma([\{x\}, \{\}, \{s,t\}, \{\}]) = \{[x, s, t, a], [x, t, s, a], [x, s, t, b], \ldots\} \]

... and its interpretation:

- Describes the set of all concrete cache states in which $x$, $s$, and $t$ occur,
  - $x$ with an age of 0,
  - $s$ and $t$ with an age not older than 2.
Sound Update – Local Consistency

Abstract Update

$\gamma$

Lifted Concrete Update

Concrete cache states

Concrete cache states

$(must) \rightarrow (must')$
LRU: Must-Analysis: Update

"Potential Cache Miss":

| {x} | {z} |
| {}  | {}  |
| {s,t}| {}  |
| {}  | {s,t}|  

"Definite Cache Hit":

| {x} | {s} |
| {}  | {x} |
| {s,t}| {t} |
| {}  | {}  |

Why does \( t \) not age in the second case?
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

```
<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>c</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{c,f}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{d}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Intersection + Maximal Age
```
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

```
\{a\} \quad \{c\} \quad \{\}
\{\} \quad \{e\} \quad \{\}
\{c,f\} \quad \{a\} \quad \{a,c\}
\{d\} \quad \{d\} \quad \{d\}
```

“Intersection + Maximal Age”
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):
- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

```
\{a\}  \{c\}  \{\}
\{\}  \{e\}  \{}
\{c,f\}  \{a\}  \{a,c\}
\{d\}  \{d\}  \{d\}
```

“Intersection + Maximal Age”
LRU: Must-Analysis: Join

Need to combine information where control-flow merges. Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a}</td>
<td></td>
<td>{c}</td>
</tr>
<tr>
<td>{c}</td>
<td></td>
<td>{e}</td>
</tr>
<tr>
<td>{a}</td>
<td>{d}</td>
<td></td>
</tr>
<tr>
<td>{d}</td>
<td></td>
<td>{d}</td>
</tr>
</tbody>
</table>
```

“Intersection + Maximal Age”
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

\[ \begin{array}{c}
\{a\} \\
\{\} \\
\{c,f\} \\
\{d\}
\end{array} \quad \sqcup \quad \begin{array}{c}
\{c\} \\
\{e\} \\
\{a\} \\
\{d\}
\end{array} \quad \begin{array}{c}
\{\} \\
\{\} \\
\{a,c\} \\
\{d\}
\end{array} \]

“Intersection + Maximal Age”
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

```
{a}
{}
{c,f}
{d}

{c}
{e}
{a}
{d}

{ }
{ }
{a,c}
{d}
```

“Intersection + Maximal Age”
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

```
<table>
<thead>
<tr>
<th></th>
<th>{a}</th>
<th>{c}</th>
<th>{}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{}</td>
<td>{e}</td>
<td>{}</td>
</tr>
<tr>
<td></td>
<td>{c,f}</td>
<td>{a}</td>
<td>{a,c}</td>
</tr>
<tr>
<td></td>
<td>{d}</td>
<td>{d}</td>
<td>{d}</td>
</tr>
</tbody>
</table>
```

“Intersection + Maximal Age”

How many memory blocks can be in the must-cache?
Example: Must-Analysis

```
entry [{}], [{}, {}, {}, {}], exit
```

A → B → D → C → A
Example: Must-Analysis

Entry: [{}, {}, {}, {}]

\[\downarrow \sqcup [{}, {}, {}, {}] = [{}, {}, {}, {}]\]

Diagram:

- Entry: [{}, {}, {}, {}]
- A
- B
- C
- D
- Exit: \[\downarrow\]
Example: Must-Analysis

\[ \text{entry} \quad [\{\}, \{\}, \{\}, \{\}, \{\}] \]

\[ \bot \sqcup [\{\}, \{\}, \{\}, \{\}, \{\}] = [\{\}, \{\}, \{\}, \{\}, \{\}] \]

\[ [\{A\}, \{\}, \{\}, \{\}, \{\}] \]

\[ [\{A\}, \{\}, \{\}, \{\}, \{\}] \]

\[ [\{A\}, \{\}, \{\}, \{\}, \{\}] \]

\[ [\{A\}, \{\}, \{\}, \{\}, \{\}] \]

entry → A

A → B

B → C

C → D

D → exit

\[ \bot \]
Example: Must-Analysis

\[\text{entry} \quad \{\}, \{\}, \{\}, \{\}, \{\}\]

\[\downarrow \sqcup [\{\}, \{\}, \{\}, \{\}] = [\{\}, \{\}, \{\}, \{\}]

\[\{A\}, \{\}, \{\}, \{\}, \{\}\]

\[\{A\}, \{\}, \{\}, \{\}, \{\}\]

\[\downarrow \sqcup [\{B\}, \{A\}, \{\}, \{\}] \sqcup [\{C\}, \{A\}, \{\}, \{\}] = [\{\}, \{A\}, \{\}, \{\}]

\[\{A\}, \{\}, \{\}, \{\}, \{\}\]

\[\{A\}, \{\}, \{\}, \{\}, \{\}\]

\[\{B\}, \{A\}, \{\}, \{\}\]

\[\{C\}, \{A\}, \{\}, \{\}\]

\[\{\}, \{A\}, \{\}, \{\}\]

\[\{\}, \{A\}, \{\}, \{\}\]

\[\downarrow \sqcup [\{\}, \{\}, \{\}, \{\}] = [\{\}, \{\}, \{\}, \{\}]

\[\text{exit} \quad \downarrow\]
Example: Must-Analysis

\[
\begin{align*}
\text{entry} & \quad [\{\}, \{\}, \{\}, \{\}, \{\}] \\
\{\{D\}, \{\}, \{A\}, \{\}\} \uplus [\{\}, \{\}, \{\}, \{\}, \{\}] & = [\{\}, \{\}, \{\}, \{\}, \{\}] \\
[\{A\}, \{\}, \{\}, \{\}, \{\}] & \quad \text{exit} \quad [\{D\}, \{\}, \{A\}, \{\}] \\
[\{B\}, \{A\}, \{\}, \{\}] \uplus [\{C\}, \{A\}, \{\}, \{\}] & = [\{\}, \{A\}, \{\}, \{\}] \\
\end{align*}
\]

No cache hits can be predicted :-(

Jan Reineke  Caches in WCET Analysis  Advanced Lecture, 2013  17 / 51
Context-Sensitive Analysis/Virtual Loop-Unrolling

- **Problem:**
  - The first iteration of a loop will always result in cache misses.
  - Similarly for the first execution of a function.
- **Solution:**
  - Virtually Unroll Loops: Distinguish the first iteration from others
  - Distinguish function calls by calling context.

Virtually unrolling the loop once:
- **Accesses to** $A$ and $D$ are provably hits after the first iteration
- **Accesses to** $B$ and $C$ can still not be classified. Within each execution of the loop, they may only miss once.
  → Persistence Analysis
LRU: May-Analysis: Abstract Domain

- Used to predict cache misses.
- Maintains lower bounds on ages of memory blocks.
- Lower bound \( \geq \) associativity

\[ \rightarrow \text{memory block definitely not cached.} \]

**Example**

<table>
<thead>
<tr>
<th>Abstract state:</th>
<th>Describes the set of all concrete cache states in which no memory blocks except ( x, y, s, t, ) and ( u ) occur,</th>
</tr>
</thead>
<tbody>
<tr>
<td>{x,y} age 0</td>
<td>( x ) and ( y ) with an age of at least 0,</td>
</tr>
<tr>
<td>{}</td>
<td>( s ) and ( t ) with an age of at least 2,</td>
</tr>
<tr>
<td>{s,t} age 3</td>
<td>( u ) with an age of at least 3.</td>
</tr>
<tr>
<td>{u}</td>
<td>( \gamma([{x, y}, {}, {s, t}, {u}]) = {[x, y, s, t], [y, x, s, t], [x, y, s, u], \ldots} )</td>
</tr>
</tbody>
</table>

\[\gamma([\{x, y\}, \{\}, \{s, t\}, \{u\}]) = \{[x, y, s, t], [y, x, s, t], [x, y, s, u], \ldots\}\]
LRU: May-Analysis: Update

“Definite Cache Miss”: 

```
{z}  ->  {s,t}
{y}  ->  {s,t}
{}   ->  {}
{x}  ->  {x}
```

“Potential Cache Hit”: 

```
{s}   ->  {s}
{y}   ->  {y,t}
{}   ->  {}
{x}  ->  {x}
```

Why does t age in the second case?
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

\[
\begin{array}{c|c|c}
\{a\} & \{c\} & \{a,c\} \\
\{\} & \{e\} & \{e\} \\
\{c,f\} & \{a\} & \{f\} \\
\{d\} & \{d\} & \{d\}
\end{array}
\]

“Union + Minimal Age”
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

```
\[
\begin{array}{c|c|c}
\{a\} & \{c\} & \{a,c\} \\
\{\}  & \{e\}  & \{e\}  \\
\{c,f\} & \{a\}  & \{f\}  \\
\{d\}  & \{d\}  & \{d\}  \\
\end{array}
\]
```

“Union + Minimal Age”
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>{}</td>
<td>{c}</td>
</tr>
<tr>
<td>{}</td>
<td>{e}</td>
<td></td>
</tr>
<tr>
<td>{c,f}</td>
<td>{a}</td>
<td>{d}</td>
</tr>
<tr>
<td>{d}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

"Union + Minimal Age"
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

```
\begin{array}{c}
\{a\} \\
\{} \\
\{c,f\} \\
\{d\}
\end{array}
\sqcup
\begin{array}{c}
\{c\} \\
\{e\} \\
\{a\} \\
\{d\}
\end{array}
= \\
\begin{array}{c}
\{a,c\} \\
\{e\} \\
\{f\} \\
\{d\}
\end{array}
```

“Union + Minimal Age”
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

```
\[
\begin{array}{c|c|c}
& \{a\} & \{c\} \\
\hline
\{\} & \{e\} & \{e\} \\
\{c,f\} & \{a\} & \{f\} \\
\{d\} & \{d\} & \{d\}
\end{array}
\]
```

"Union + Minimal Age"
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures $\gamma$ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

```
\[
\begin{array}{c}
\{a\} \\
\{c,f\} \\
\{d\} \\
\{\}
\end{array}
\sqcup
\begin{array}{c}
\{c\} \\
\{e\} \\
\{a\} \\
\{d\}
\end{array}
= 
\begin{array}{c}
\{a,c\} \\
\{e\} \\
\{f\} \\
\{d\}
\end{array}
\]
```

“Union + Minimal Age”