On the Correctness, Optimality and Precision of Static Probabilistic Timing Analysis

Sebastian Altmeyer, Rob Davis

Date 2014 - Dresden
Random Cache Replacement Policy - Why

- in general: yet another cache replacement policy
- for real-time systems: randomize execution time to apply a probabilistic analysis [1]

![Execution time distribution](image1.png)

![Exceedance function](image2.png)

Execution time distribution

Exceedance function
Random Cache Replacement Policy - How

Using evict-on-miss policy [4], on a cache miss:

- randomly select and replace a cache way
- each way is selected with equal probability \((1/N)\) \((N = \text{associativity})\)

Simplifying assumptions:

- fully associative cache (extension to set-associative trivial)
- analysis of one trace only (aka. single path program)
- constant hit/miss delays
Given the following access sequence

\[ a \rightarrow b \rightarrow c \rightarrow d \rightarrow a \]

what is the probability that the second access to \( a \) is cache hit?

\[
P(a \text{ is cache hit}) = P(b \text{ does not replace } a) \cdot P(c \text{ does not replace } a) \cdot P(d \text{ does not replace } a)
\]

\[
= \left( \frac{N-1}{N} \right) \cdot \left( \frac{N-1}{N} \right) \cdot \left( \frac{N-1}{N} \right)
\]

\[
= \left( \frac{N-1}{N} \right)^3
\]

In general

\[ P(e^{hit}) = \left( \frac{N - 1}{N} \right)^k \]

where \( k \) is the reuse distance and \( N \) the associativity.

Reuse distances \( \hat{=} \) \# of intervening accesses.

Example:

\[ a \rightarrow b \rightarrow a^1 \rightarrow c \rightarrow d \rightarrow b^3 \rightarrow c^2 \rightarrow f \rightarrow a^5 \rightarrow c^2 \]
1. Compute reuse distances $k$ (# of intervening accesses): 

$$a \rightarrow b \rightarrow a \rightarrow c \rightarrow d \rightarrow b \rightarrow c \rightarrow f \rightarrow a \rightarrow c$$

2. Derive hit-probabilities[5] for all accesses $e$ with reuse distance $k$:

$$P(e^{hit}) = \left( \frac{N - 1}{N} \right)^k$$

3. Derive probability mass functions:

$$\mathcal{I}_i = \begin{pmatrix} \text{hit-delay} & \text{miss-delay} \\ P(e^{hit}) & P(e^{miss}) \end{pmatrix}$$

4. Compute convolution $\otimes$ to derive execution-time distribution:

$$p\text{WCET} = \otimes \mathcal{I}_i$$
Analysis of Random Caches: Example of Convolution

Reuse distances
\[ a \rightarrow b \rightarrow a^1 \rightarrow c \rightarrow d \rightarrow b^3 \rightarrow c^2 \rightarrow f \rightarrow a^5 \rightarrow c^2 \]

Probability mass functions
\[
\begin{pmatrix}
H & M \\
0 & 1
\end{pmatrix} \otimes \begin{pmatrix}
H & M \\
0 & 1
\end{pmatrix} \otimes \begin{pmatrix}
H & M \\
3/4 & 1/4
\end{pmatrix} \otimes \begin{pmatrix}
H & M \\
0 & 1
\end{pmatrix} \otimes \begin{pmatrix}
H & M \\
0 & 1
\end{pmatrix} \otimes \begin{pmatrix}
H & M \\
3/4 & 1 - (3/4)^3
\end{pmatrix} \otimes
\]

Resulting miss-distribution
\[
\begin{pmatrix}
4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
0 & 0.0237 & 0.153 & 0.337 & 0.324 & 0.139 & 0.0210 & 0
\end{pmatrix}
\]

Exceedance function

On the Correctness, Optimality and Precision of Static Probabilistic Timing Analysis
Analysis of Random Caches - Is it correct?

Well ...

$$P(e^{hit}) = \left( \frac{N - 1}{N} \right)^k$$  \hspace{1cm} (1)

is not the exact hit-probability, but a lower bound. ✓

Convolution requires independence ...
but hits/misses in a random cache are not independent. ✗
Consider a cache with associativity $N = 2$ and the access sequence:

$$a \rightarrow b \rightarrow c \rightarrow a^2 \rightarrow b^2 \rightarrow c^2$$

Equation (1) allows up to 3 hits:

$$P(e_{a^2}^{hit}) \cdot P(e_{b^2}^{hit}) \cdot P(e_{c^2}^{hit}) = \left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^2 > 0$$

but only 2 are possible (pigeon-hole principle):

$$P(\#\text{hits} = 3) = P(e_{a^2}^{hit} \land e_{b^2}^{hit} \land e_{c^2}^{hit}) = 0$$
Can we fix the analysis?

Another cache-hit probability [2]

\[ \hat{P}^D(e^{hit}) = \begin{cases} \left( \frac{N-1}{N} \right)^k & k < N \\ 0 & \text{otherwise} \end{cases} \]  \hspace{1cm} (2)

and yet another cache-hit probability [3]

\[ \hat{P}^K(e^{hit}) = \left( \frac{N - 1}{N} \right) \sum P(e_j^{miss}) \]  \hspace{1cm} (3)

where \( \sum P(e_j^{miss}) \) is the sum over the probabilities of misses of intervening memory accesses.
Can we fix the analysis?

Another cache-hit probability [2]

\[
\hat{P}^D(e_{hit}) = \begin{cases} 
(N-1)^k/N & k < N \\
0 & \text{otherwise}
\end{cases} 
\]  

(2)

and yet another cache-hit probability [3]

\[
\hat{P}^K(e_{hit}) = \left( \frac{N-1}{N} \right) \sum P(e_{j_{miss}}) 
\]  

(3)

where \( \sum P(e_{j_{miss}}) \) is the sum over the probabilities of misses of intervening memory accesses.

Which equation is correct, let alone optimal? What does correct mean?
Random Cache Replacement Policy

Correctness and Optimality
   Correctness Conditions
   Optimality
   Improvement via Cache Contention

Cache State Enumeration

Evaluation

Conclusions
Correctness

**Problem:** Computing the exact hit-probability of an access $e$, $P(e^{hit})$, is hard and hits/misses are not independent.

**Solution:** Define an independent lower bound $\hat{P}(e^{hit})$ on the exact hit-probability $P(e^{hit})$
Correctness

**Problem:** Computing the **exact hit-probability** of an access \( e \), \( P(e^{hit}) \), is hard and hits/misses are **not independent**.

**Solution:** Define an **independent lower bound** \( \hat{P}(e^{hit}) \) on the exact hit-probability \( P(e^{hit}) \)

Such a **lower bound** is correct, if

(i) \( \forall e \in [e_1, \ldots, e_n] : P(e^{hit}) \geq \hat{P}(e^{hit}) \),

and it is **independent** of other accesses, if

(ii) \( \forall E \subseteq [e_1, \ldots, e_n] : P(\land_{e \in E} e^{hit}) \geq \prod_{e \in E} \hat{P}(e^{hit}) \).
Which equation is correct?

<table>
<thead>
<tr>
<th>Equation</th>
<th>(i)</th>
<th>(ii)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{P}(e^{hit}) = \left( \frac{N-1}{N} \right)^k )</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>( \hat{P}^D(e^{hit}) = \begin{cases} \left( \frac{N-1}{N} \right)^k &amp; k &lt; N \ 0 &amp; \text{otherwise} \end{cases} )</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>( \hat{P}^K(e^{hit}) = \left( \frac{N-1}{N} \right) \sum P(e_j^{miss}) )</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

(i) \( \forall e \in [e_1, \ldots, e_n] : P(e^{hit}) \geq \hat{P}(e^{hit}) \),

(ii) \( \forall E \subseteq [e_1, \ldots, e_n] : P(\bigwedge_{e \in E} e^{hit}) \geq \prod_{e \in E} \hat{P}(e^{hit}) \).
Optimal Approximation?

Claim: Only the reuse distance $k$
(and the associativity $N$ of the cache) is required for
static probabilistic analysis of random replacement

Proof sketch:
For any $k$ and any $N$, we can construct an access sequence where
$\hat{P}_D(e_{\text{hit}})$ cannot be improved:

Case $k < N$:
$[e_1, e_2, e_3, \ldots, e_{k-1}, e_k, e_1, e_2, e_3, \ldots, e_{k-1}, e_k]$
(all $e_i$, $e_j$ are pairwise distinct)
Optimal Approximation?

**Claim:** Only the reuse distance $k$ 
(and the associativity $N$ of the cache) is required for 
static probabilistic analysis of random replacement

If so, then

$$\hat{P}_D(e^{\text{hit}}) = \begin{cases} 
\left(\frac{N-1}{N}\right)^k & k < N \\
0 & \text{otherwise}
\end{cases}$$

is optimal.
Optimal Approximation?

**Claim:** Only the reuse distance \( k \) (and the associativity \( N \) of the cache) is required for static probabilistic analysis of random replacement.

If so, then

\[
\hat{P}^D(e_{hit}) = \begin{cases} 
\left( \frac{N-1}{N} \right)^k & k < N \\
0 & \text{otherwise}
\end{cases}
\]

is optimal.

**Proof sketch:**
For any \( k \) and any \( N \), we can construct an access sequence where \( \hat{P}^D \) can not be improved:

Case \( k < N \): \([e_x, e_1, e_2, e_3, \ldots e_{k-1}, e_k, e_x]\)

Case \( k \geq N \): \([e_1, e_2, e_3, \ldots e_{k-1}, e_k, e_1, e_2, e_3, \ldots e_{k-1}, e_k]\)

(all \( e_i, e_j \) are pairwise distinct)
Optimal, yes. But precise?

\[
\hat{P}^D(e^{hit}) = \begin{cases} 
\left(\frac{N-1}{N}\right)^k & k < N \\
0 & \text{otherwise}
\end{cases}
\]

1. Reuse distance \( k \) is an upper bound and not the actual number of cache misses.
2. Unable to predict any hits if \( k \geq N \).

Consider the access sequence (with \( N = 4 \))

\[
a \rightarrow b \rightarrow c \rightarrow d \rightarrow f \rightarrow a^4 \rightarrow b^4 \rightarrow c^4 \rightarrow d^4 \rightarrow f^4
\]

where \( \hat{P}^D(e^{hit}) = 0 \) holds for each access.
Improvement via Cache Contention

- \( \hat{P}^D(e^{hit}) \) is set to 0 to ensure independence constraint \((N \leq k)\)
- Yet, we can do so without setting each probability to 0

The cache contention is defined as:

\[
\text{con} \hat{=} \# \text{ of last accesses considered to be cached}
\]

The equation is:

\[
\hat{P}^N(e^{hit}) = \begin{cases} 
\left( \frac{N-1}{N} \right)^k & \text{if } \text{con}(e_l, T) < N \\
0 & \text{otherwise}
\end{cases}
\]

<table>
<thead>
<tr>
<th>rd</th>
<th>a, b, c, d, f, a, b, c, d, f</th>
</tr>
</thead>
<tbody>
<tr>
<td>con</td>
<td>0, 0, 0, 0, 0, 1, 2, 3, 4, 3</td>
</tr>
<tr>
<td>( \hat{P}^N )</td>
<td>0, 0, 0, 0, 0, ( \left( \frac{3}{4} \right)^4 ), ( \left( \frac{3}{4} \right)^4 ), ( \left( \frac{3}{4} \right)^4 ), 0, ( \left( \frac{3}{4} \right)^4 )</td>
</tr>
</tbody>
</table>
Remark:

Equation

\[ \hat{P}^N(e_{hit}) = \begin{cases} \left(\frac{N-1}{N}\right)^k & \text{con}(e_i, T) < N \\ 0 & \text{otherwise} \end{cases} \]

uses the

- reuse distance \( k \),
- associativity \( N \),
- order of the accesses.

Drawbacks:

- no re-ordering allowed
- degrades composability of the analysis
Random Cache Replacement Policy

Correctness and Optimality

Correctness Conditions
Optimality
Improvement via Cache Contention

Cache State Enumeration

Evaluation

Conclusions
Alternative Idea: Exhaustive Cache State Enumeration

**Idea:** Simply compute all cache states on a given trace.

Completely orthogonal approach to convolution, no approximation.

- Single cache state: \((\text{Cache Content}, \text{Probability})\)
- Set of cache states: \(P(\text{Cache Content}, \text{Probability})\)
- Sound assumption: start with an empty cache.
Example

Example sequence:

\[ a \rightarrow b \rightarrow a \rightarrow c \rightarrow d \rightarrow \ldots \]

(with associativity \( N = 4 \))

- Single cache state: \((\text{Cache Content}, \text{Probability})\)
- Empty cache: \((\emptyset, 1)\)

\((\emptyset, 1)\)
Example

Example sequence:

\[ a \rightarrow b \rightarrow a \rightarrow c \rightarrow d \rightarrow \ldots \]

(with associativity \( N = 4 \))

- Single cache state: \((Cache \ Content, Probability)\)
- Empty cache: \((\emptyset, 1)\)

\[ \begin{array}{c}
\emptyset \rightarrow 1 \\
\{a\} \rightarrow 1 \\
(a, 1) \rightarrow 1 \\
\end{array} \]
Example

Example sequence:

\[ a \rightarrow b \rightarrow a \rightarrow c \rightarrow d \rightarrow \ldots \]

(with associativity \( N = 4 \))

- Single cache state: \((\text{Cache Content}, \text{Probability})\)
- Empty cache: \((\emptyset, 1)\)

```
\[
\begin{array}{c}
\text{a} \\
\text{b}
\end{array}
\begin{array}{c}
(\emptyset, 1) \\
(\{a\}, 1) \\
(\{a, b\}, 3/4) \\
(\{b\}, 1/4)
\end{array}
\]
```

On the Correctness, Optimality and Precision of Static Probabilistic Timing Analysis
Example

Example sequence:

\[ a \rightarrow b \rightarrow a \rightarrow c \rightarrow d \rightarrow \ldots \]

(with associativity \( N = 4 \))

- Single cache state: \((\text{Cache Content}, \text{Probability})\)
- Empty cache: \((\emptyset, 1)\)

\[ \begin{array}{c}
\text{a} \\
\downarrow \quad 1 \\
\text{a} \\
\downarrow \quad \frac{3}{4}
\end{array} \]

\[ \begin{array}{c}
\text{b} \\
\downarrow \quad \frac{3}{4} \\
\{a, b\}, \frac{3}{4}
\end{array} \hspace{1cm} \begin{array}{c}
\text{b} \\
\downarrow \quad \frac{1}{4} \\
\{b\}, \frac{1}{4}
\end{array} \]

\[ \begin{array}{c}
\text{a} \\
\downarrow \quad \frac{3}{4} \\
\{a, b\}, \frac{15}{16}
\end{array} \hspace{1cm} \begin{array}{c}
\text{a} \\
\downarrow \quad \frac{3}{16} \\
\{a\}, \frac{1}{16}
\end{array} \]
Example

Example sequence:

\[ a \rightarrow b \rightarrow a \rightarrow c \rightarrow d \rightarrow \ldots \]

(with associativity \( N = 4 \))

- Single cache state: (Cache Content, Probability)
- Empty cache: (\( \emptyset \), 1)

```
          (\( \emptyset \), 1)
            ↓
          (\{a\}, 1)
            ↓
          (\{a, b\}, \frac{3}{4}) (\{b\}, \frac{1}{4})
            ↓
          (\{a, b\}, \frac{15}{16}) (\{a\}, \frac{1}{16})
            ↓
          (\{a, b, c\}, \frac{15}{32}) (\{b, c\}, \frac{15}{64}) (\{a, c\}, \frac{18}{64}) (\{c\}, \frac{1}{64})
```

On the Correctness, Optimality and Precision of Static Probabilistic Timing Analysis
Complexity and number of states?

- once a memory block was cached, it still may be cached.
  ⇒ up to $2^l$ states (where $l$ is the number of memory blocks)
  ⇒ exhaustive enumeration is computationally intractable
Two orthogonal approaches so far ...

1. Analysis using convolution
   ▶ fast
   ▶ imprecise
2. Analysis using state enumeration
   ▶ slow
   ▶ precise

Can we combine both?
Combined Analysis

1. Identify set of **relevant memory blocks**
   (# of occurrences used to indicate relevance)
   # relevant blocks = trade-off parameter (precision vs. runtime)

2. **Split trace** into two subtraces
   (relevant subtrace, non-relevant subtrace)

3. **analyse** both traces **independently**

4. combine results (convolution)
Example

Assume access sequence:

\[ a \rightarrow b \rightarrow a \rightarrow c \rightarrow d \rightarrow b \rightarrow c \rightarrow f \rightarrow a \rightarrow c \]

with two relevant blocks \( \{a, c\} \)

Compute

▷ precise probability distribution \( D_P \) for the sequence

\[ a \rightarrow \Box \rightarrow a \rightarrow c \rightarrow \Box \rightarrow \Box \rightarrow c \rightarrow \Box \rightarrow a \rightarrow c \]

▷ fast calculation of \( D_F \) for the sequence

\[ \Box \rightarrow b \rightarrow \Box \rightarrow \Box \rightarrow d \rightarrow b \rightarrow \Box \rightarrow f \rightarrow \Box \rightarrow \Box \]

Combination:

\[ D_{\text{complete}} = D_F \otimes D_P \]
Random Cache Replacement Policy

Correctness and Optimality
  Correctness Conditions
  Optimality
  Improvement via Cache Contention

Cache State Enumeration

Evaluation

Conclusions
Evaluation – Binary Search

Exceedance Function

132 memory accesses in total, 25 distinct memory blocks, associativity 16
Discrete-cosine transformation on a 8x8 pixel block. 2185 memory accesses in total, 347 distinct memory blocks, associativity 16.
Discrete-cosine transformation on a 8x8 pixel block. 2185 memory accesses in total, 347 distinct memory blocks, associativity 16.

More examples in the technical report
Random Cache Replacement Policy

Correctness and Optimality
  Correctness Conditions
  Optimality
  Improvement via Cache Contention

Cache State Enumeration

Evaluation

Conclusions
Conclusions

Analysis of random cache replacement using:

- Convolution
  - Correctness conditions
  - Classification of existing approaches
  - Optimality shown for Equation (2)
  - New approach using cache contention

- Exhaustive state enumeration

- Combined approach (state enum. + convolution)
Conclusions

Analysis of random cache replacement using:

- Convolution
  - Correctness conditions
  - Classification of existing approaches
  - Optimality shown for Equation (2)
  - New approach using cache contention
- Exhaustive state enumeration
- Combined approach (state enum. + convolution)

Future work:

- improve hit-probability by considering other information
- extend analysis to control-flow graphs
Conclusions

Analysis of random cache replacement using:

- Convolution
  - Correctness conditions
  - Classification of existing approaches
  - Optimality shown for Equation (2)
  - New approach using cache contention
- Exhaustive state enumeration
- Combined approach (state enum. + convolution)

Future work:

- Improve hit-probability by considering other information
- Extend analysis to control-flow graphs

Thank you for your attention!
Bibliography


**Exceedance function (Counterexample)**

\[ a \rightarrow b \rightarrow c \rightarrow a^2 \rightarrow b^2 \rightarrow c^2 \]

**Correct miss-distribution**

\[
\begin{pmatrix}
3 & 4 & 5 & 6 & 7 \\
0 & 0.125 & 0.5625 & 0.3125 & 0
\end{pmatrix}
\]

**Miss-distribution with Equation (1)**

\[
\begin{pmatrix}
3 & 4 & 5 & 6 & 7 \\
0.015625 & 0.140625 & 0.421875 & 0.421875 & 0
\end{pmatrix}
\]

**Exceedance function**
Example Combined Approach (Precise Computation $D_P$)

\[
\begin{align*}
a &\rightarrow \square \rightarrow a \rightarrow c \rightarrow \square \rightarrow \square \rightarrow c \rightarrow \square \rightarrow a \rightarrow c
\end{align*}
\]
Example Combined Approach (Fast Computation $D_F$)

\[ \square \rightarrow b \rightarrow \square \rightarrow \square \rightarrow d \rightarrow b^3 \rightarrow \square \rightarrow f \rightarrow \square \rightarrow \square \]

Probability mass functions

\[
\begin{pmatrix} H & M \end{pmatrix} \otimes \begin{pmatrix} H & M \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} H & M \\ \frac{3}{4} & 1 - \frac{3}{4} \end{pmatrix} \otimes \begin{pmatrix} H & M \end{pmatrix}
\]

Resulting miss-distribution

\[
\begin{pmatrix} 2 & 3 & 4 & 5 \\ 0 & 0.421875 & 0.578125 & 0 \end{pmatrix}
\]
Example Combined Approach (Resulting Distributions)

\[ \begin{align*}
    a \rightarrow b & \rightarrow a \rightarrow c \\
    a \rightarrow d & \rightarrow b \rightarrow c \\
    a \rightarrow f & \rightarrow a \rightarrow c
\end{align*} \]

Precise Computation 1-CDF (#misses)

Fast Computation 1-CDF (#misses)

Combined Results 1-CDF (#misses)

\[ D_{\text{complete}} = D_F \otimes D_P \]
Cache Contention

\[ con: \mathbb{E} \times \mathbb{T} \rightarrow \mathbb{N} \]

\[ con(e_l, [e_1, e_2, \ldots, e_{l-1}]) = \]
\[ \left| \left\{ e_i \in [e_1, \ldots, e_{l-1}] \mid (l - rd(e_l, [e_1, \ldots, l - 1]) < i \land \hat{P}(e_i^{hit}) \neq 0) \right\} \right| \]
\[ \cup \left\{ e_r \in [e_1, \ldots, e_{l-1}] \mid (l - rd(e_l, [e_1, \ldots, l - 1]) = r \right\} \right| \quad (4) \]
Evaluation – insertion-sort

Exceedance Function

707 memory accesses in total, 21 distinct memory blocks, associativity 16
Automatically generated code (statemate). 1831 memory accesses in total, 394 distinct memory blocks, associativity 16.
Simulation of an extended Petri Net (nsichneu). 5202 memory accesses in total, 454 distinct memory blocks, associativity 16.
Finite impulse response filter (fir). 3419 memory accesses in total, 393 distinct memory blocks, associativity 16.