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

Sebastian Altmeyer, Rob Davis

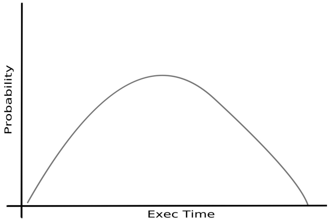
Date 2014 - Dresden

University of Amsterdam

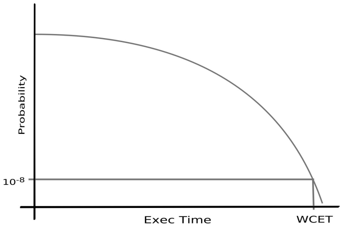
University of York

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



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

Analysis of Random Caches: Cache-Hit Probability

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?

$$\begin{aligned} P(\text{a is cache hit}) = \quad & P(b \text{ does not replace } a) \\ & \cdot P(c \text{ does not replace } a) \\ & \cdot P(d \text{ does not replace } a) \\ = \quad & \left(\tfrac{N-1}{N}\right) \cdot \left(\tfrac{N-1}{N}\right) \cdot \left(\tfrac{N-1}{N}\right) \\ = \quad & \left(\tfrac{N-1}{N}\right)^3 \end{aligned}$$

Analysis of Random Caches: Cache-Hit Probability [5]

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:

$$a \rightarrow b \rightarrow a^1 \rightarrow c \rightarrow d \rightarrow b^3 \rightarrow c^2 \rightarrow f \rightarrow a^5 \rightarrow c^2$$

Analysis of Random Caches in a Nutshell [4]

1. Compute **reuse distances** k (# of intervening accesses):

$$a \rightarrow b \rightarrow a^1 \rightarrow c \rightarrow d \rightarrow b^3 \rightarrow c^2 \rightarrow f \rightarrow a^5 \rightarrow c^2$$

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** $\bigotimes$ to derive execution-time distribution:

$$pWCET = \bigotimes \mathcal{I}_i$$

Analysis of Random Caches: Example of Convolution

Reuse distances

$$a \rightarrow b \rightarrow a^1 \rightarrow c \rightarrow d \rightarrow b^3 \rightarrow c^2 \rightarrow f \rightarrow a^5 \rightarrow c^2$$

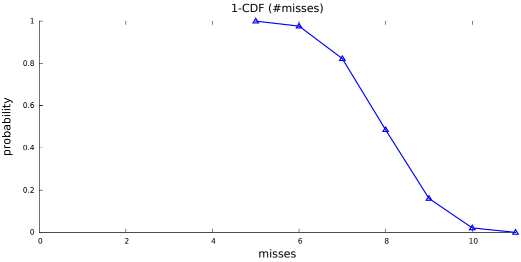
Probability mass functions

$$\begin{array}{ccccccccccccc} a & \rightarrow & b & \rightarrow & a^1 & \rightarrow & c & \rightarrow & d & \rightarrow & b^3 & \rightarrow \ldots \\ \begin{pmatrix} H & M \\ 0 & 1 \end{pmatrix} & \otimes & \begin{pmatrix} H & M \\ 0 & 1 \end{pmatrix} & \otimes & \begin{pmatrix} H & M \\ \frac{3}{4} & \frac{1}{4} \end{pmatrix} & \otimes & \begin{pmatrix} H & M \\ 0 & 1 \end{pmatrix} & \otimes & \begin{pmatrix} H & M \\ 0 & 1 \end{pmatrix} & \otimes & \begin{pmatrix} H & M \\ (\frac{3}{4})^3 & 1-(\frac{3}{4})^3 \end{pmatrix} & \otimes \end{array}$$

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



Analysis of Random Caches - Is it correct?

Well ...

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

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

Convolution **requires independence** ...
but hits/misses in a random cache are **not independent**. ✗

Analysis of Random Caches - Counterexample

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:

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

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

$$P(\#\text{hits} = 3) = P(e^{hit}_{a^2} \wedge e^{hit}_{b^2} \wedge e^{hit}_{c^2}) = 0$$

Can we fix the analysis?

Another cache-hit probability [2]

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

and yet another cache-hit probability [3]

$$\hat{P}^{K}(e^{hit}) = \left(\frac{N-1}{N}\right)^{\sum P(e_j^{\text{miss}})} \tag{3}$$

where $\sum P(e_j^{\text{miss}})$ is the sum over the probabilities of misses of intervening memory accesses.

Can we fix the analysis?

Another cache-hit probability [2]

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

and yet another cache-hit probability [3]

$$\hat{P}^K(e^{hit}) = \left(\frac{N-1}{N}\right)^{\sum P(e_j^{\text{miss}})} \quad (3)$$

where $\sum P(e_j^{\text{miss}})$ 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) $\forall E \subseteq [e_1, \ldots, e_n] : P\left(\bigwedge_{e \in E} e^{hit}\right) \geq \prod_{e \in E} \hat{P}(e^{hit})$.

Which equation is correct?

	(i)	(ii)
$\hat{P}(e^{hit}) = \left(\frac{N-1}{N}\right)^k$	✓	✗
$\hat{P}^D(e^{hit}) = \begin{cases} \left(\frac{N-1}{N}\right)^k & k < N \\ 0 & \text{otherwise} \end{cases}$	✓	✓
$\hat{P}^K(e^{hit}) = \left(\frac{N-1}{N}\right)^{\sum P(e_j^{\text{miss}})}$	✗	✗

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

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

Optimal Approximation?

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

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**.

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

cache contention **con** $\hat{=}$
\# of last accesses considered to be cached

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

	a,	b,	c,	d,	f,	a,	b,	c,	d,	f
rd	∞	∞	∞	∞	∞	4	4	4	4	4
con	0	0	0	0	0	1	2	3	4	3
$\hat{P}^N$	0	0	0	0	0	$(\frac{3}{4})^4$	$(\frac{3}{4})^4$	$(\frac{3}{4})^4$	0	$(\frac{3}{4})^4$

Remark:

Equation

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

uses the

- reuse distance k,
- associativity N,
- **order of the accesses**.

Drawbacks:

- no re-ordering allowed
- degrades composability of the analysis

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: $(\mathit{Cache\ Content}, \mathit{Probability})$
- Set of cache states: $\mathcal{P}(\mathit{Cache\ Content}, \mathit{Probability})$
- Sound assumption: start with an empty cache.

Example

Example sequence :

$$a \rightarrow b \rightarrow a \rightarrow c \rightarrow d \rightarrow \dots$$

(with associativity $N = 4$)

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

a

$$\begin{tikzcd} (\emptyset, 1) \arrow[d, dashed, "1"] \\ (\{a\}, 1) \end{tikzcd}$$

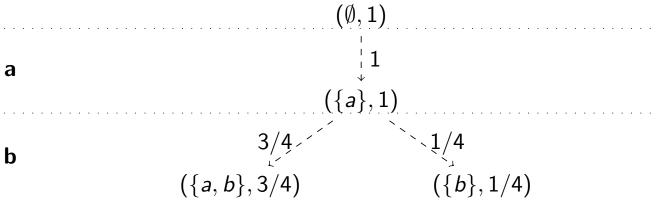
Example

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: $(\emptyset, 1)$



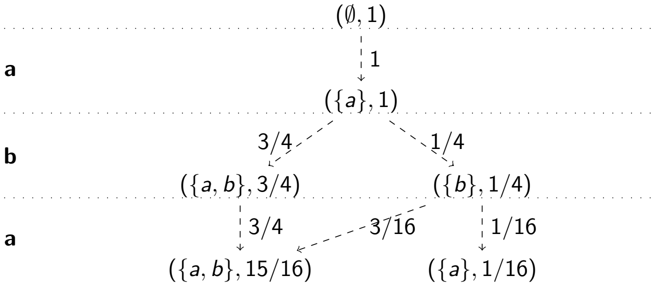
Example

Example sequence :

$$a \rightarrow b \rightarrow a \rightarrow c \rightarrow d \rightarrow \dots$$

(with associativity $N = 4$)

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



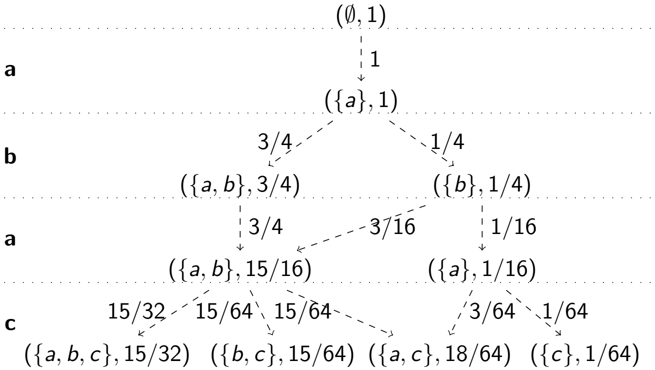
Example

Example sequence :

$$a \rightarrow b \rightarrow a \rightarrow c \rightarrow d \rightarrow \dots$$

(with associativity $N = 4$)

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



Complexity and number of states?

- once a memory block was cached, it still may be cached.

$\Rightarrow$ up to 2^l states (where l is the number of memory blocks)

$\Rightarrow$ 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:

$$a \to b \to a \to c \to d \to b \to c \to f \to a \to c$$

with two **relevant blocks** $\{a, c\}$

Compute

- precise probability distribution D_P for the sequence

$$a \to \square \to a \to c \to \square \to \square \to c \to \square \to a \to c$$

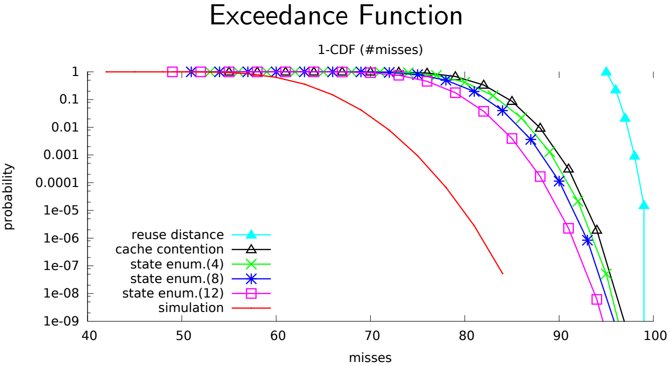
- fast calculation of D_F for the sequence

$$\square \to b \to \square \to \square \to d \to b \to \square \to f \to \square \to \square$$

Combination:

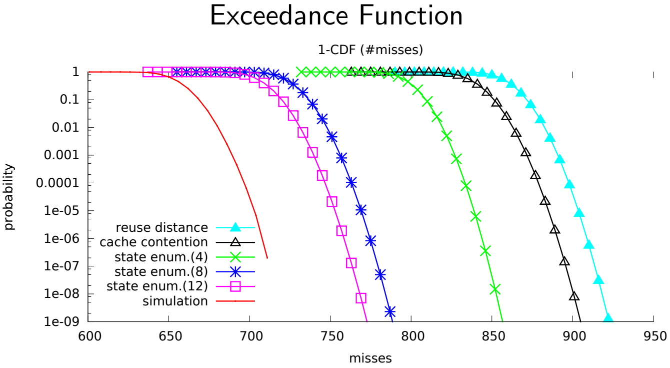
$$D_{complete} = D_F \otimes D_P$$

Evaluation – Binary Search



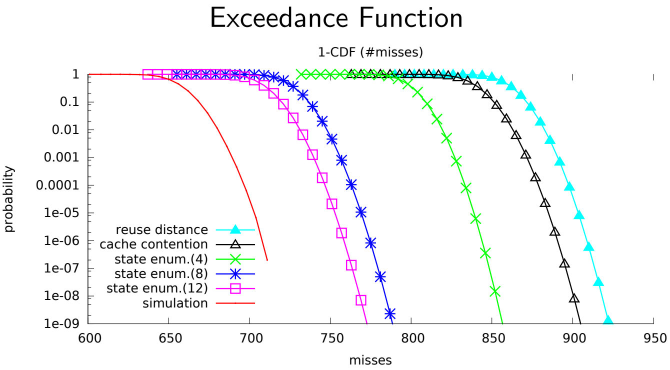
132 memory accesses in total, 25 distinct memory blocks, associativity 16

Evaluation – Jfdctint (gem5-trace)



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

Evaluation – Jfdctint (gem5-trace)



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
`www.cs.york.ac.uk/ftpdir/reports/2013/YCS/487/YCS-2013-487.pdf`

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

[1] F. J. Cazorla, E. Quiñones, T. Vardanega, L. Cucu, B. Triquet, G. Bernat, E. D. Berger, J. Abella, F. Wartel, M. Houston, L. Santinelli, L. Kosmidis, C. Lo, and D. Maxim.
Proartis: Probabilistically analyzable real-time systems.
ACM Trans. Embedded Comput. Syst., 12(2s):94, 2013.

[2] R. Davis, L. Santinelli, S. Altmeyer, C. Maiza, and L. Cucu-Grosjean.
Analysis of probabilistic cache related pre-emption delays.
In *ECRTS 2013*, pages 129–138, 2013.

[3] L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla.
A cache design for probabilistically analysable real-time systems.
In *DATE 2013*, pages 513–518, 2013.

[4] E. Quinones, E. D. Berger, G. Bernat, and F. J. Cazorla.
Using randomized caches in probabilistic real-time systems.
In *ECRTS 2009*, pages 129–138, 2013.

[5] S. Zhou.
An efficient simulation algorithm for cache of random replacement policy.
In *NPC 2010*, pages 144–154, 2010.

Exceedance function (Counterexample)

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

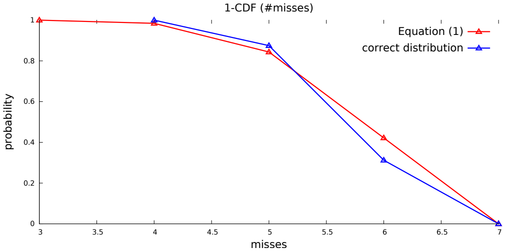
Correct miss-distribution

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

Miss-distribution with Equation (1)

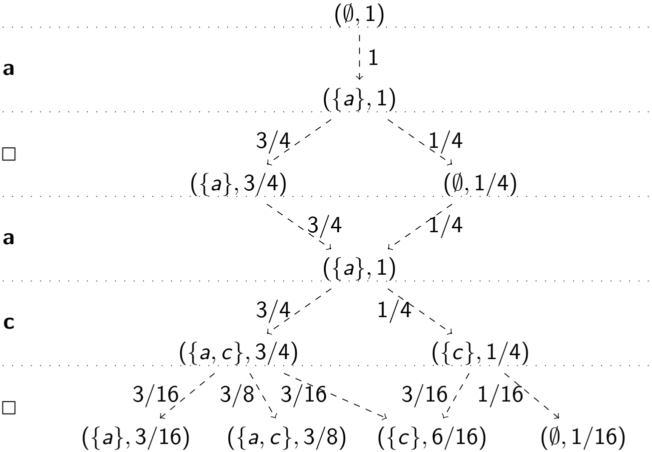
$$\begin{pmatrix} 3 & 4 & 5 & 6 & 7 \\ 0.015625 & 0.140625 & 0.421875 & 0.421875 & 0 \end{pmatrix}$$

Exceedance function



Example Combined Approach (Precise Computation D_P)

$$a \to \square \to a \to c \to \square \to \square \to c \to \square \to a \to c$$



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

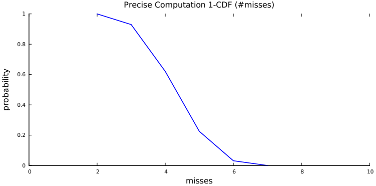
$$\begin{array}{ccccccc} b & \dashrightarrow & d & \rightarrow & b^3 & \dashrightarrow & f \\ \begin{pmatrix} H & M \\ 0 & 1 \end{pmatrix} & \otimes & \begin{pmatrix} H & M \\ 0 & 1 \end{pmatrix} & \otimes & \begin{pmatrix} H & M \\ \frac{3}{4}^3 & 1 - \frac{3}{4}^3 \end{pmatrix} & \otimes & \begin{pmatrix} H & M \\ 0 & 1 \end{pmatrix} \end{array}$$

Resulting miss-distribution

$$\begin{pmatrix} 2 & 3 & 4 & 5 \\ 0 & 0.421875 & 0.578125 & 0 \end{pmatrix}$$

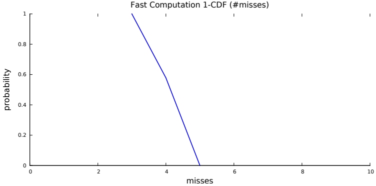
Example Combined Approach (Resulting Distributions)

$a \rightarrow \square \rightarrow a \rightarrow c \rightarrow \square \rightarrow \square \rightarrow c \rightarrow \square \rightarrow a \rightarrow c$



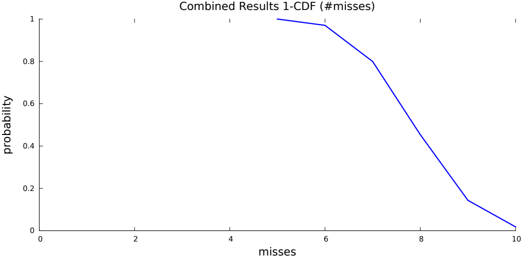
D_P

$\square \rightarrow b \rightarrow \square \rightarrow \square \rightarrow d \rightarrow b \rightarrow \square \rightarrow f \rightarrow \square \rightarrow \square$



D_F

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



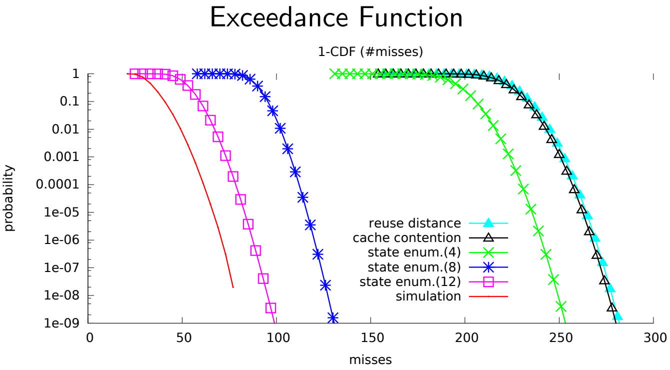
$D_{complete} = D_F \otimes D_P$

Cache Contention

$$con \colon \mathbb{E} \times \mathbb{T} \to \mathbb{N}$$

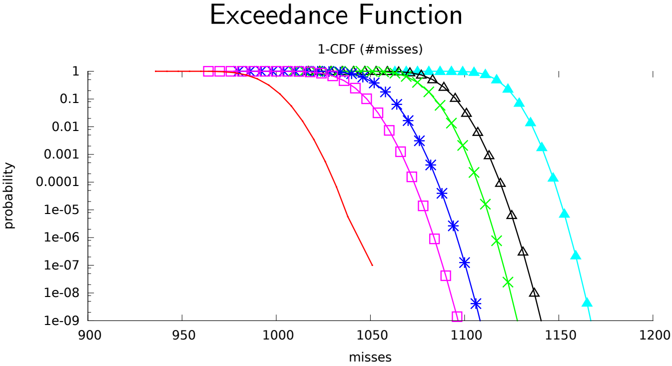
$$\begin{aligned} & con\,(e_l, [e_1, e_2, \ldots, e_{l-1}]) = \\ & \Big| \Big\{ e_i \in [e_1, \ldots, e_{l-1}] \;\Big|\; \Big(l - rd(e_l, [e_1, \ldots, l-1]) < i \wedge \hat{P}(e_i^{hit}) \neq 0 \Big) \Big\} \\ & \qquad \cup \Big\{ e_r \in [e_1, \ldots, e_{l-1}] \;\Big|\; (l - rd(e_l, [e_1, \ldots, l-1]) = r \Big\} \Big| \end{aligned} \quad (4)$$

Evaluation – insertion-sort



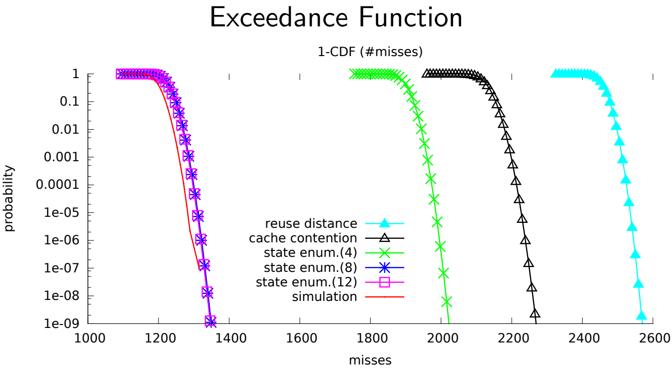
707 memory accesses in total, 21 distinct memory blocks, associativity 16

Evaluation – statemate (gem5-trace)



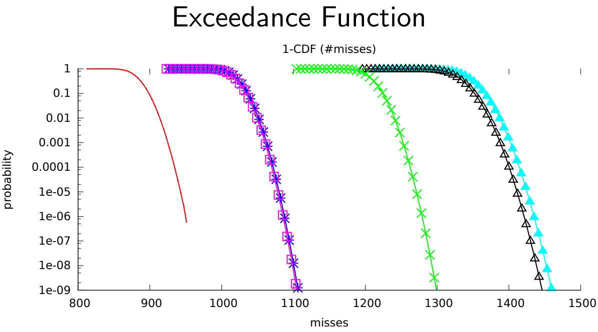
Automatically generated code (statemate). 1831 memory accesses in total, 394 distinct memory blocks, associativity 16.

Evaluation – nsichneu (gem5-trace)



Simulation of an extended Petri Net (nsichneu). 5202 memory accesses in total, 454 distinct memory blocks, associativity 16.

Evaluation – fir (gem5-trace)



Finite impulse response filter (fir). 3419 memory accesses in total, 393 distinct memory blocks, associativity 16.