Analysis of Probabilistic Cache Related Pre-emption Delays

Rob Davis, Luca Santinelli, Sebastian Altmeyer, Claire Maiza, and Liliana Cucu-Grosjean
Outline

- Probabilistic real-time systems
- System model
  - Random cache replacement policies (Evict-on-Access, Evict-on-Miss)
- Static Probabilistic Timing Analysis (SPTA)
  - Single path programs
  - Complexity
- Cache Related Pre-emption Delays
  - At a specific point
  - Upper bounding the effect at any point
  - Multiple pre-emption
- Extension to Multi-path programs
- Evaluation
  - Case study and simulation
- Conclusions and future work
Probabilistic Real-Time Systems

- What do we mean by a probabilistic real-time system?
  - One or more parameters are described by random variables
  - Example: instead of a single WCET value, we have a probabilistic Worst-Case Execution Time (pWCET)
  - Characterised by a probability distribution

Common question: What does this mean?
Isn’t WCET defined as the single worst-case execution time value?
Rolling 10 dice (only interested in how many sixes)

- WCET equates to 10 sixes
- pWCET upper bound probability distribution on number of sixes rolled

What should the budget for ‘sixes’ be such that we get an expected failure rate no higher than 1 per 1 million rolls of the set of dice? i.e. runs of the program. (Failure = more sixes than budgeted)
Common misunderstanding: Difference between pET and pWCET

- Analogy: two options
  - 10x ordinary dice
  - 3x big dice that show pairs of values e.g. 2 sixes at once
  - Like a program with two paths

- Different pETs for the two options (typically dependent)
- pWCET is a tight upper bound on all possible pETs (independent)
- pWCETs can be composed to get pWCRTs
Static Probabilistic Timing Analysis (SPTA)

- Aim is to show that the probability of timing failure falls below some threshold e.g. $10^{-9}$ failures per hour: $p_{WCET}$ v. budget

![Diagram showing CPU, Instruction Cache, Memory, and Probabilistic WCET (pWCET) distribution]
Cache model

- Fully associative instruction cache of $N$ blocks
  - Memory blocks can be loaded into any block in cache
  - Each instruction resides in a memory block
  - Memory blocks may contain multiple instructions

- Instruction modelling
  - When an instruction is requested its memory block may be in cache (a hit) or not (a miss)
  - If it is not in cache, then it has to be fetched from main memory and loaded into the cache.
  - On a miss, a random location is chosen in the cache to accommodate the new memory block (Evict-on-Miss random replacement policy)
  - Each cache block has the same probability of being evicted $1/N$
Evict-on-miss random replacement

Cache with memory blocks a, b, c, d, e loaded next instruction is in memory block f
Instruction modelling

- Instructions are either:
  - Cache hit or cache miss (when executed)

- Program path
  - Is a sequence of instructions
  - Represented by the sequence of memory blocks for those instructions e.g. \( a, b, a, c, d, b, c, d, a, e, b, f, e, g, a, b, h \)

- Re-use distance \( k \)
  - Defined as the maximum possible number of evictions since the last access to the memory block containing the required instruction
    \( a, b, a^1, c, d, b^3, c^2, d^2, a^5, e, b^4, f, e^2, g, a^5, b^4, h \)
  - Can have re-use distance of zero (instructions in the same block & EoM)
    \( a, a^0, b, b^0, b^0, b^0, a^1, \)
Each instruction has a probability of being a cache hit or a cache miss:
- Described by a discrete random variable (PMF)

\[
I = \begin{pmatrix}
H \\ P\{\text{hit}\}
M \\ P\{\text{miss}\} = 1 - P\{\text{hit}\}
\end{pmatrix}
\]

Note $H$ and $M$ are times for a cache Hit and cache Miss

Example:
- Probability of a cache hit = 0.75 with an execution time of 1
- Probability of a cache miss = 0.25 with an execution time of 10

\[
I = \begin{pmatrix}
1 & 10 \\ 0.75 & 0.25
\end{pmatrix}
\]

For each instruction we aim to lower bound the probability of a cache hit independent of whether previous instructions were hits or misses
Probabilistic real-time analysis

- Requires independence:
  - Two random variables $X$ and $Y$ are independent if they describe two events such that the outcome of one event does not have any impact on the outcome of the other
  - In our context an instruction having a particular execution time is an event
  - There is a dependency between these events via the cache

- Key idea is to conservatively model the execution times of instructions as independent random variables (which have no dependency on whether previous instructions were cache hits or cache misses)

- Actual probability of a cache hit $P\{hit\}$ is dependent on the outcome of previous events (hits or misses) but we lower bound it with $P_{hit}$ which is independent then we can use convolution to get $p_{WCET}$ distribution for a sequence of instructions
Probabilistic real-time analysis

- Summation of independent random variables is via convolution

\[ C_j = \mathcal{I}_1 \otimes \mathcal{I}_2 \otimes \ldots, \]

\[ P\{Z = z\} = \sum_{k=-\infty}^{+\infty} P\{X_1 = k\}P\{X_2 = z - k\} \]

\[
\begin{pmatrix}
1 & 10 \\
0.8 & 0.2
\end{pmatrix} \otimes \begin{pmatrix}
1 & 10 \\
0.7 & 0.3
\end{pmatrix} = \begin{pmatrix}
2 & 11 & 20 \\
0.56 & 0.38 & 0.06
\end{pmatrix}
\]
Static Probabilistic Timing Analysis (SPTA)

- Sequence of instructions represented by their memory blocks and re-use distances
  a, b, a¹, c, d, b³, c², d², a⁵, e, b⁴, f, e², g, a⁵, b⁴, h
- Evict-on-miss random replacement policy
  - (Recall: Fully associative cache, N cache blocks, on a cache miss we randomly choose a cache block to be evicted)
  - Initial analysis by Zhou [17] 2010

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

- Depends only on re-use distance \( k \) (not on actual cache hit / miss behaviour)

**Formulation is not strictly correct due to a dependency via the finite size of the cache**
Problem of Independence

- **Counter example:**
  - Consider a cache of size $N = 2$
    - $a, b, c, b^1, a^3,$
  - If the 2nd access to $b$ is a hit, then $b$ and $c$ must be in cache at that point and so the 2nd access to $a$ is **certain** to be a miss

Probability that the 2nd access to block $a$ is a hit is **not independent** of whether previous instructions were hits or misses

Joint probability that 2nd accesses to both $a$ and $b$ are hits is zero, not 1/16 (as obtained from Zhou formula and convolution)

- **Solution:**
  - Need to model instruction PMFs as independent (so can we can compose using convolution)

**HOW?**
Upper bound the maximum amount of known information ($h$ blocks that could be known to be in cache) and consider how this may reduce the effective cache size and number of possible evictions
Static Probabilistic Timing Analysis

- Evict-on-Miss
  - With \( h \) intervening hits assumed (if \( h \geq N \) then \( P_{hit} = 0 \))
    \[
P_{hit}^E(k, h) = \left( \frac{N - h - 1}{N - h} \right)^{k-h}
    \]
  - Lower bound (for all values of \( h \)) so crucially independent of previous hits / misses
    \[
P_{EoM}^hit(k) = \begin{cases} 
    \left( \frac{N-1}{N} \right)^k & k < N \\
    0 & k \geq N
  \end{cases}
    \]
  - Similarly for Evict-on-Access (Cucu-Grosjean et al. [6])
    \[
P_{EoA}^hit(k) = \begin{cases} 
    \left( \frac{N-(k-1)-1}{N-(k-1)} \right)^k & k < N \\
    0 & k \geq N
  \end{cases}
    \]

Proof in the paper

Easy to see that Evict-on-Miss dominates Evict-on-Access
Static Probabilistic Timing Analysis

- Upper bound pWCET for each instruction based on re-use distance \( k \) using formula modelling independent (lower bound) probability of a cache hit
- pWCET for a single path by convolution \( C_j = I_1 \otimes I_2 \otimes \ldots \),
  - Convolution is commutative and associative
  - Can represent a sequence of accesses
    \[ a, b, a^1, c, d, b^3, c^2, d^2, a^5, e, b^4, f, e^2, g, a^5, b^4, h \]
  by their re-use distances:
  \[ Q = \{-, -, 1, -, -, 3, 2, 2, 5, -, 4, -, 2, -, 5, 4, -\} \]
  \[ Q^{PROG} = \{1, 2, 2, 2, 3, 4, 4, 5, 5, -, -, -, -, -, -, -, -, -\} \]
pWCET distribution (1-CDF)

pWCET without pre-emption

1e-09

143
Complexity of SPTA

- Convolving pWCETs for $n$ instructions
  - Might seem to have exponential complexity $O(2^n)$
    (The case if each distribution had two arbitrary values)
  - Max value is a small constant $M$ so after $n$ convolutions, max value is $nM$ and $2nM$ operations are required for the $(n+1)$th convolution
  - Complexity is pseudo-polynomial $O(Mn^2)$ where $M$ is a small constant

Problem is tractable in practice

Can also use re-sampling to reduce the size of the distributions
Effects of pre-emption at a single specific program point
- Pre-emption assumed to flush the cache making some re-use distances infinite
- Pre-emption after 1st access
  \[ a, b, a^1, c, d, b^3, c^2, d^2, a^5, e, b^4, f, e^2, g, a^5, b^4, h \]
  \[ Q_1 = \{1\} \]
- Pre-emption after 5th access
  \[ a, b, a^1, c, d, b^3, c^2, d^2, a^5, e, b^4, f, e^2, g, a^5, b^4, h \]
  \[ Q_5 = \{2, 2, 3, 5\} \]

Accounting for effects of pre-emption
- Remove values from representation of program (path)
  \[ Q^{PROG} = \{1, 2, 2, 2, 3, 4, 4, 5, 5, -, -, -, -, -, -, -, -\} \]
  \[ Q^{PROG}_{P_5} = \{1, 2, 4, 4, 5, -, -, -, -, -, -, -, -, -, -, -, -\} \]
Effects of a single pre-emption at any program point

- Concept of a dominant virtual pre-emption point with an impact that upper bounds the impact of pre-emption at any actual program point.

Method to create virtual pre-emption point $P^*$

- Pad representations of pre-emption effects so they are all the same length e.g. $Q_1 = \{1, -, -, -, -\}$ and $Q_5 = \{2, 2, 3, 5\}$.

- Apply $\min^+(Q_i, Q_j) = \{k_r = \min(k_{i,r}, k_{j,r}) \forall r \leq |Q_i|\}$.

- So $\min^+(Q_1, Q_5) = \{1, 2, 3, 5\}$.

- Do this for all possible pre-emption points:
  
  $Q^* = \min^+_{r \in \{1, \ldots, 16\}} \{Q_r\} = \{1, 2, 3, 5\}$.

- Remove values from representation of program (path)

  $Q_{P^*}^{PROG} = \text{pre}(Q_{P^*}^{PROG}, Q^*)$

  $Q_{P^*}^{PROG} = \{2, 2, 4, 4, 5, -, -, -, -, -, -, -, -, -, -, -, -\}$
pWCET distribution (1-CDF)

- pWCET without pre-emption
- pWCET with 1 pre-emption

Execution time:
- 143
- 170
pCRPD: Multiple pre-emption

- Effects of **multiple** pre-emptions
  - Remove values multiple times
    \[ Q^{PROP}_{xP^*} = pre(Q^{PROP}_{(x-1)P^*}, Q^*) \]
  - If a specific value is no longer present (this is due to pessimism in the analysis) remove next larger value (don’t remove smaller ones)
  - Example: \( a, b, c, d, a^3, b^3, c^3, d^3, d^0, d^0, d^0, d^0, d^0, d^0 \)
    \[ Q^* = \{0, 3, 3, 3\} \]
    \[ Q^{PROP}_{4P^*} = \{0, 0, -, -, -, -, -, -, -, -, -, -, -\} \]
  - 4 pre-emptions are not enough to force this program to all misses
Multi-path Programs

- Static Probabilistic Timing Analysis (SPTA) and pCRPD extended to multi-path programs
  - SPTA intuition
    - Upper bound re-use distances using program analysis (fixed point iteration)
    - Combine & collapse sub-paths to get a synthetic path representation that upper bounds the pWCET of any path through the program
  - pCRPD intuition
    - Upper bound the pre-emption effect at each program point using program analysis (on the re-use distances obtained by SPTA before collapsing)
    - Combine effects for all program points into a single dominant virtual pre-emption point P*
    - Apply P* to synthetic path as in the single path case

Details in the paper
Evaluation

- Used Malardalen Benchmarks
  - FAC, FIBCALL, FDCT, JFDCINT (single path with loops)
  - BS, INSERTSORT, FIR (multi-path)
  - Compared Evict-on-Miss and Evict-on-Access random replacement policies
- Varied:
  - Number of pre-emptions
  - Cache size ($N = 256, 128, 64, 32$)
  - Memory block sizes ($1, 2, 4, 8$ instructions)
- Assumed $H = 1, M = 10$
- Also compared SPTA and pCRPD analysis with simulation
FAC Benchmark

- Evict on Miss
- Memory block size = 1
- Cache size $N = 128$

Probability

pWCET estimate from simulation $10^7$ runs

pWCET without pre-emption

4 pre-emption

3 pre-emption

2 pre-emption

1 pre-emption

Execution time
FAC Benchmark

- Evict on Miss
- Memory block size = 1
- Cache size N = 128
FAC Benchmark

- Evict on Access
- Memory block size = 1
- Cache size N = 128

Simulation much closer due to evictions on every access

Worse performance than Evict-on-Miss
FAC Benchmark
- Evict on Miss
- Memory block size = 4
- Cache size N = 128
FAC Benchmark

- Evict on Access
- Memory block size = 4
- Cache size N = 128
FAC Benchmark

- Evict on Miss
- Varying memory block size
- Cache size $N = 128$

![Graph showing execution time and probability for different block sizes (bs)]
FAC Benchmark

- Evict on Miss
- Varying memory block size
- and cache size

![Graph showing probability over execution time with varying block sizes and cache sizes.]

- PROG bs=1 N=256
- PROG bs=2 N=128
- PROG bs=4 N=64
- PROG bs=8 N=32
Technical Report

YCS-2012-477

- Find it on Rob Davis publications page:
  http://www-users.cs.york.ac.uk/~robdavis/publications.html

Results for other benchmarks

- FAC is very simple code. Others require many more pre-emptions to reduce them to all misses (e.g. > 500 pre-emptions for INSERTSORT)
Conclusions

Main contributions
- Revised Static Probabilistic Timing Analysis for Evict-on-Miss random cache replacement policy
  - Fixed a problem with dependency
- Extended SPTA to multipath programs
- Introduced analysis of pCRPD
  - Including multiple pre-emptions of multi-path programs
- Evaluations
  - Method is feasible and provides results that give a useful upper bound on the pWCET

Future work
- Improvements to the pWCET analysis via loop un-rolling
- Comparisons with deterministic analysis for systems with traditional cache replacement policies
- Reduce the pessimism in SPTA
Questions?