Improving Time-Randomized Cache Designs

Pedro Benedicte[†,‡], Carles Hernandez[†], Jaume Abella[†], Francisco J. Cazorla[†,⋆]
pbenedic@bsc.es, carles.hernandez@bsc.es, jaume.abella@bsc.es, francisco.cazorla@bsc.es
[†] Barcelona Supercomputing Center (BSC) [‡] Universitat Politècnica de Catalunya (UPC) [⋆] IIIA-CSIC

***Abstract*—Enabling timing analysis for caches has been pursued by the critical real-time embedded systems (CRTES) community for years due to their potential to reduce worst-case execution times (WCET). Measurement-based protabilistic timing analysis (MBPTA) techniques have emerged as a solution to time-analyze complex hardware including caches, as long as they implement some random policies. Existing random placement and replacement policies have been proven efficient to some extent for single-level caches. However, they may lead to some probabilistic pathological eviction scenarios. In this work we propose new random placement and replacement policies specifically tailored for multi-level caches and for avoiding any type of pathological case.**

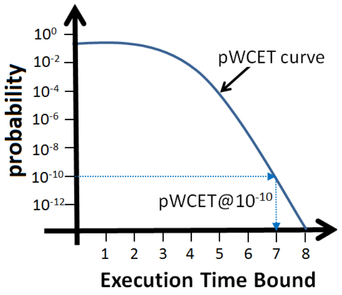

Fig. 1. Example of pWCET distribution.

I. Introduction

WCET estimation for real-time software is needed for the certification of critical systems against safety standards. WCET estimates need to be reliable and as tight as possible. A common misconception is that a WCET estimate overrun necessarily causes a system level failure. However, this is not true since mandatory safety measures are in place to manage sporadic faults. Following the probabilistic approach used to handle random hardware faults [5], MBPTA reasons on WCET as a distribution, aka probabilistic WCET (pWCET) curve (Figure 1), describing the maximum probability with which a WCET estimate can be exceeded.

MBPTA builds on a set of measurements taken during system analysis phase. Those measurements are passed as input to Extreme Value Theory (EVT) [9], a statistical tool to estimate an upper-bound distribution for distribution tails (high execution times in our case). MBPTA imposes how execution time measurements must be collected so that they capture those conditions that lead to execution times matching or upper-bounding those during system operation. EVT, part of MBPTA, requires that the execution times meet several statistical properties related to the degree of independence and identical distribution of the random variable (execution times) modelled, and whether it can be modelled with an exponential tail, which is the most convenient distribution for pWCET estimates of real-time programs [3].

Hardware time randomized caches enable an efficient application of MBPTA. They implement random placement and replacement techniques. Currently, one replacement and two placement MBPTA-compliant policies have been proposed:

Conventional Random Replacement (CRR) [8]: makes random eviction choices so that, in the event of a miss in a given set, for a cache with W ways, the probability of a line in that set to be evicted is $1/W$. CRR builds on a pseudo-random number generator (PRNG) with sufficient quality to allow cache conflicts to be truly random.

hash Random Placement (hRP) [7]: uses a parametric hash function whose input includes the memory address to be accessed and a random seed. It produces the (random) set where the address is placed with that random seed. Thus, whether two addresses are placed or not in the same set is a random event. Any two addresses can be placed in the same set with a probability $1/S$, where S is the number of sets. Upon change of the random seed, addresses are randomly and independently mapped into sets.

Random Modulo placement (RM) [4]: Unlike hRP, RM placement preserves the advantages in terms of spatial locality as modulo placement does. In particular, RM prevents conflicts between cache lines close enough in memory, as modulo placement (MOD) does, but still providing random placement as needed by MBPTA. This is achieved by randomly permuting the location of cache lines within a memory segment using a random seed. Upon a random seed change, addresses in a segment are randomly permuted, thus leading to random placement across segments and no conflicts within segments.

However, CRR may produce probabilistic pathological cases with relevant probabilities, and hRP and RM do not provide efficient randomization for multi-level caches.

II. Random Cache Replacement

A. Proposal

Conventional random replacement (CRR) is the most suitable replacement policy for MBPTA due to its probabilistic nature: replacement choices are random and independent. CRR makes pathological replacement patterns probabilistic rather than systematic, though they can still occur. We propose Random Permutations Replacement (RPR) [2], that limits pathological random replacement scenarios by increasing temporal reuse and enforcing random evictions to occur across all cache ways.

- When accessed data fits in a cache set, they will eventually be placed in different cache lines, thus avoiding potentially long mutual evictions by construction.
- When the number of accessed lines exceeds the size of a set, RPR effects are also positive increasing reuse, though the impact of replacement naturally reduces.

To reach its goals, RPR leverages the concept of random permutations [6].

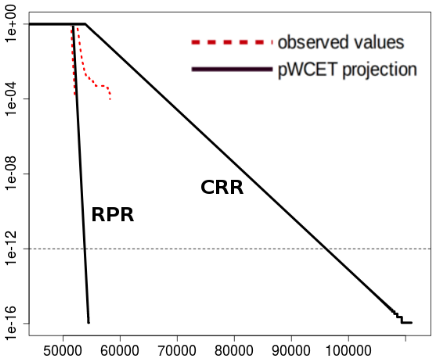

Fig. 2. pWCET curve for `jfdc-tint` Mälardalen benchmark.

B. Results

Figure 2 shows the pWCET distribution when using both CRR and RPR for `jfdctint` Mälardalen Benchmark. Red dotted lines and black straight lines represent the CCDF for the measured data and the pWCET curves respectively. RPR provides increasingly higher gains as the exceedance threshold decreases due to the fact that RPR avoids pathological evictions by construction. Since CRR can produce some such pathological evictions with relevant probability, MBPTA accounts for that by smoothening the shape of the curve and shifting it to the right.

III. Random Cache Placement

A. Proposal

hRP and RM have been proposed for single-level cache systems. For multi-level cache systems, the desired characteristics are the following: (1) **WCET reduction** as the main metric to optimize. (2) **Reduced impact on average performance** due to the importance of this metric for mixed-critical scenarios executing tasks with different criticality levels. (3) **Preserve time composability** by breaking the dependence of WCET estimates on the actual memory addresses used, to favor incremental software development. (4) **MBPTA compliance** to reduce the cost of changing existing timing analysis tools.

Simple solutions consisting of applying just hRP or RM to each cache level do not comply with all characteristics to a sufficient extent. Our proposed solution [1] (shown in Figure 3), uses RM in the L1 and a combination of hRP (at page level) and Modulo (inside each page) in the L2. This design meets all the desired characteristics, while offering a reasonable implementation cost and complexity.

B. Results

We implemented the proposed design in an FPGA and run the EEMBC benchmarks. For all benchmarks we observe (Figure 4) that the pWCET estimate is above the High Water Mark (HWM) (as expected) and for all benchmarks but one the pWCET estimate with hRP+MOD is below HWM+20% (industrial practice for deterministic techniques) obtained for MOD. Hence, hRP+MOD helps reducing WCET estimates w.r.t. current practice while increasing the confidence on estimates w.r.t. just increasing the HWM by a fudge factor of 20%. On average, pWCET estimates are only 8% above the HWM (12 percentage points below HWM+20%).

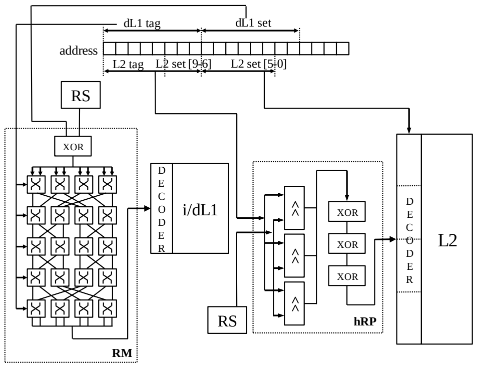

Fig. 3. Multi-level random cache design.

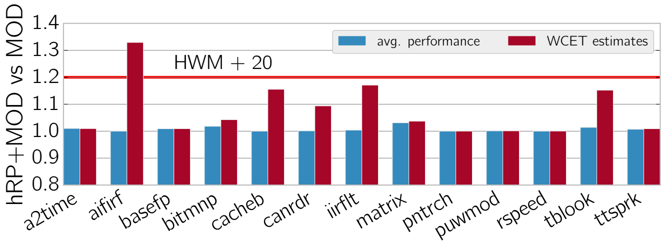

Fig. 4. MOD average results (left) and pWCET hRP+MOD results (right).

IV. Conclusions and Future Work

Existing time-randomized caches help CRTES achieve higher performance while also making them time analyzable. However, they are not sufficiently efficient for multi-level caches and do not avoid pathological cases. Hence, we proposed new MBPTA-compliant placement and replacement policies overcoming those limitations.

Our future work consists of enabling the use of further high-performance computing (HPC) hardware solutions for CRTES in the context of MBPTA. For instance, our next target consists of enabling data prefetching in CRTES.

References

[1] P. Benedicte, C. Hernandez, J. Abella, and F. J. Cazorla. Design and integration of hierarchical-placement multi-level caches for real-time systems. In *DATE*, 2018.

[2] P. Benedicte, C. Hernandez, J. Abella, and F. J. Cazorla. Rpr: A random replacement policy with limited pathological replacements. In *SAC*, 2018.

[3] L. Cucu-Grosjean et al. Measurement-based probabilistic timing analysis for multi-path programs. In *ECRTS*, 2012.

[4] C. Hernandez et al. Random modulo: a new processor cache design for real-time critical systems. In *DAC*, 2016.

[5] IOS. *ISO/DIS 26262. Road Vehicles – Functional Safety*, 2009.

[6] J. Jalle, L. Kosmidis, J. Abella, E. Quinones, and F. Cazorla. Bus designs for time-probabilistic multicore processors. In *DATE*, 2014.

[7] L. Kosmidis et al. A cache design for probabilistically analysable real-time systems. In *DATE*, 2013.

[8] L. Kosmidis et al. Probabilistic timing analysis on conventional cache designs. In *DATE*, 2013.

[9] S. Kotz et al. *Extreme value distributions: theory and applications*. World Scientific, 2000.

Pedro Benedicte is a PhD. Student for the CAOS group at BSC. He obtained his M.S. degree in 2016 and graduated in Informatics Engineering in 2014, both titles obtained from the Universitat Politècnica de Catalunya. His current research focuses on the computer architecture of Real-Time Systems, more specifically, on increasing guaranteed performance in those systems by using techniques commonly found in High Performance Computers as well as Probabilistic Timing Analysis.