On the application of Static Probabilistic Timing Analysis to Memory Hierarchies [1]

Benjamin Lesage[1], David Griffin[1], Robert I. Davis[1], Sebastian Altmeyer[2]
[1] University of York, UK / [2] University of Amsterdam, The Netherlands
{benjamin.lesage,david.griffin,rob.davis}@york.ac.uk, altmeyer@uva.nl

I. Introduction

The temporal validation of a real-time system must ensure that critical tasks meet their deadline, even in the worst-case scenarios. Traditionally, both the deadline and worst-case execution time of a task are expressed as absolute bounds. Such absolutes may lead to resource over-provisioning for the sake of system validation, as they may include unlikely events with a granularity beyond the industry requirements. Probabilistic worst-case execution times (pWCET) fit industrial requirements for the validation of critical real-time systems as they provide bounds on the worst-case execution time of applications as well as their exceedance probability when run. Different static probabilistic timing analyses (SPTA) [2], [3], [1] have been proposed and extended to derive the probabilistic worst-case execution time (pWCET) of a task in the presence of randomised caches.

However, these methods only apply to single level caches and fail to capture the behaviour of complex memory hierarchies. Such hierarchies are common even in simple architectures as they bridge the gap between fast processors and relatively slower main memories. In the context of a memory hierarchy, multiple cache layers are linked together. They are traversed in order from the processor to the main memory, and a request is served by the first cache level where the target block is found. Therefore, the position of a requested block in the hierarchy defines the latency of a memory request, and the behaviour of subsequent requests.

Static analyses have been proposed to upper-bound the contribution of cache hierarchies on tasks' execution times; however, these approaches focus on deterministic policies [4], [6]. Deterministic policies do not lend themselves to the expression of worst-case timing estimates as exceedance functions, and the resulting absolute bounds may be pessimistic. The prior methods rely either on the classification of the access behaviour of each access with regards to each cache level for an incremental analysis level by level [4], or a unified model of the whole hierarchy [6].

II. SPTA for Single Caches

Two complementary families of SPTA approaches, contention and collection, have been defined for single caches implementing an evict-on-miss random cache replacement policy. Under such a policy, when a requested block is absent from the cache, a line is randomly selected for eviction and the requested block, fetched from memory, replaces its contents. Each line has the same probability $\frac{1}{N}$, depending on the associativity N of the cache, to be selected for eviction, hence minimising the dependencies of the cache behaviour on access history.

Contention-based methods approximate the Probability Mass Function (PMF) of each access, i.e. the probability of the access to suffer a cache hit or a memory hit latency, such that their convolution results in a sound timing estimate. The convolution operator imposes restrictions on the derived hit probabilities. The PMF of an instruction must lower-bound its cache hit probability, and hence its execution time. As should the convolution of different PMF lower-bound the execution time of the corresponding sequence of accesses [1]. The former is ensured by the definition of a lower bound on hit probability based on the reuse distance of accesses, the maximum number of evictions from the previous access to the same block. The latter is guaranteed in the most recent approach [1] using cache contention, which relates to the number of hits between two accesses, to ensure that the lines that are considered as hits fit into the cache.

Collection approaches instead rely on the approximation of a sound subset of the possible cache states at runtime, alongside their corresponding execution time distributions. Compared to contention-based methods they offer an increased precision at the cost of higher complexity. The complexity can be controlled by focusing the analysis on a subset of memory blocks deemed relevant, which in turn introduces uncertainty in the model and reduces its precision.

III. Impact of memory hierarchies on SPTA

Consider an $M+1$ level memory hierarchy, including the main memory. A memory request is first processed by the lowest cache level the $L1$ cache. In the case of a miss, it is then transferred to the next layer and so on until it can be served. The only guarantee after the access is the presence of the block in the $L1$ cache. The contents of each layer after an access depends on the management policy. Each policy may introduce new dependencies between instructions and cache levels. Hence, the introduction of multiple levels in the memory hierarchy challenges the assumptions behind existing SPTA techniques.

For both contention and collection approaches, the hierarchy management policies need to be studied i) to prevent the explosion of the problem, e.g. from a single state an eviction on $L1$ and $L2$ produces $N_1 \times N_2$ states assuming N_i is the associativity of cache level i, and ii) capture scenarios for which convolution produces only valid sequences of events, e.g. a hit in the $L2$ cache may depend on a prior access being a miss in $L1$.

A. Inclusive cache hierarchy

An inclusive cache hierarchy enforces the inclusion of the contents of a cache in the higher levels, e.g. each block in the L1 cache must also be in the L2 cache. When a block is evicted from a high cache level, it is removed from lower levels, a so-called *inclusion victim*. The higher level caches must be able to hold all lines in the respective lower levels.

After a memory request, the block is guaranteed to be present in all levels inclusive of the first one. The reuse distance of an access can be estimated from the closest prior reference to the same block. Further, because of *inclusion victims*, a miss on cache level L contributes to the reuse distance of all lower levels. Thus an access for LM may contribute to the reuse distance of accesses on the $L1$,..., $LM-1$ caches. This is a worst-case scenario assuming a memory hit. The contribution to the reuse distance on the $L1$ of a guaranteed level L hit cannot exceed $L-1$. But randomised caches hardly provide such guarantees on hits beyond the first level. Deterministic architectures rely on history-dependent policies, not amenable in the context of SPTA, to reduce the impact of *inclusion victims* [5].

B. Exclusive cache hierarchy

An exclusive cache hierarchy maximises the use of the cache space. Every cache line in the hierarchy holds a different block. A memory block can only appear in one cache level. Upon eviction from a cache level L, the victim is inserted in cache level $L+1$. The requested data is only inserted in the $L1$ cache upon a miss. Therefore, all caches in an exclusive hierarchy must use the same line size to allow for line swapping. Although this is not mandatory, an equal number of sets across the different levels eases the implementation and the analysis.

An exclusive cache hierarchy may be modelled as a single cache which size equals the size of the hierarchy. Given a hierarchy of LRU caches, the logical age of a block then defines the cache level in which it resides and therefore its access latency [4]. A similar model can be applied to collection analyses for randomised caches, but the depth of a block in the hierarchy needs to be upper-bounded as it defines the latency of accesses. If different levels of the hierarchy use a different number of cache sets, the unified contents model may not hold.

The reuse distance could also be computed using this unified contents representation. A block can only be evicted from its current cache level if an access is served by a higher level than the one where it resides. A victim block is inserted in the higher cache, where it is less sensitive to evictions. This reinforces the need for a per-level expression of the reuse-distance, even though it is likely to be the same for most levels. Capturing guaranteed hits, requests which do not reach higher levels is complex for layers beyond the first.

C. Non/Mostly-inclusive cache hierarchy

Neither the exclusion or inclusion restriction applies on a non-inclusive hierarchy. Upon a miss on a cache level, the request is passed to the next level until it can be served. An access guarantees the insertion of the data in all the levels where it could not be found, up to the first hit in the hierarchy. This is the simplest implementation of a cache hierarchy. From the analysis point of view, it means that guarantees about the presence of data in cache after an access only exist for the first level, or if misses can be guaranteed. Deriving sound PMF requires a lower bound on the miss probabilities of accesses, e.g. an access can only be a $L3$ hit if it misses on all lower levels.

There is no correlation between the different levels. Contrary to the other policies it is unsafe to consider that a piece of data is in higher levels than the one it might reside in. This has been identified in the deterministic case, where the insertion of a block in a level which is not guaranteed to be accessed results in optimistic timing estimates. Instead, when the access to a level cannot be guaranteed, a collection approach can insert a placeholder for the data in cache, such that further accesses to the same block do not cause additional evictions, but cannot assume hits on this data. An important property to capture for memory blocks is the lowest and highest level where they might reside in the hierarchy. These two values bound the cache levels where the eviction of blocks is bound to occur but not the insertion.

It is currently an open problem how to effectively and efficiently perform SPTA for any of the above multilevel memory hierarchies using random replacement caches.

ACKNOWLEDGMENT

This work was funded by COST Action IC1202 (TACLe), and the EU FP7 Integrated Project PROXIMA (611085).

REFERENCES

[1] Sebastian Altmeyer and Robert I Davis. On the correctness, optimality and precision of static probabilistic timing analysis. In 17^{th} *DATE*, 2014.

[2] Francisco Cazorla, Eduardo Quiñones, Tullio Vardanega, Liliana Cucu, Benoit Triquet, Guillem Bernat, Emery Berger, Jaume Abella, Franck Wartel, Michael Houston, Luca Santinelli, Leonidas Kosmidis, Code Lo, and Dorin Maxim. Proartis: Probabilistically analysable real-time systems. *Transactions on Embedded Computing Systems*, 2013.

[3] Robert I Davis, Luca Santinelli, Sebastian Altmeyer, Claire Maiza, and Liliana Cucu-Grosjean. Analysis of probabilistic cache related pre-emption delays. In *25th ECRTS*, 2013.

[4] Damien Hardy and Isabelle Puaut. Wcet analysis of instruction cache hierarchies. *Journal of Systems Architecture*, 57(7), 2011.

[5] Aamer Jaleel, Eric Borch, Malini Bhandaru, Simon C. Steely Jr., and Joel Emer. Achieving non-inclusive cache performance with inclusive caches: Temporal locality aware (tla) cache management policies. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '43, pages 151–162, 2010.

[6] T. Sondag and H. Rajan. A more precise abstract domain for multi-level caches for tighter wcet analysis. In *IEEE 31st RTSS*, 2010.