RETROSPECTIVE: High Performance Cache Replacement Using Re-Reference Interval Prediction (RRIP)

Aamer Jaleel[†], Kevin B. Theobald[‡], Simon C. Steely Jr.[§], Joel Emer[†*]
[†]NVIDIA [‡]Oregon PERS [§]Intel [*]MIT

I. Overview

At the time of this ISCA 2010 paper, growing on-chip last-level cache (LLC) sizes and multi-core processors had revived cache management as a hot research topic in industry and academia. This work was a collaboration between Intel's product and research teams. Intel was then following its "tick-tock" paradigm, alternating architecture re-designs with process shrinks, and product teams always sought new features to give adequate performance gains per release.

Kevin Theobald was a Performance Architect on Intel's Oregon design team while Aamer Jaleel, Simon Steely, and Joel Emer were part of Intel's VSSAD, an advanced R&D group. Both teams were independently seeking ways to improve LLC performance. This work is the result of combining their efforts that outperformed LRU at a lower cost. The design went into Ivy Bridge [1] and later processors, and has since influenced cache research and industry-wide cache designs.

II. RRIP Development Timeline

A. Increasing Replacement State Bits

The Oregon product team had previously developed Nehalem and its process-shrink follow-on (Westmere). Theobald designed the replacement policy for Nehalem's LLC using NRU replacement [O1][1], an approximation of LRU using only 1 bit (the "age bit") per cacheline. When Oregon began work on Haswell, a future processor, Theobald sought a better age-based approximation of LRU with up to n bits per line ($n <= 4$ for our 16-way LLC), and experimented with giving new fills lower priority than hits.

[1][O#] indicates a reference in the original paper.

In parallel, the research group had been working on solutions for commonly-occurring access patterns that performed poorly under LRU (see Figure 1). The first scenario is for *thrashing access patterns* where the working set is larger than the cache. For such patterns, LRU replacement always leads to cache misses. The second scenario is for *mixed access patterns* where a frequently-referenced working set (that fits in the LLC) is intermixed with references to non-temporal data, called *scans*. For mixed access patterns, successive references to the frequently-referenced working set *before* a scan will hit in the cache, but *after* a scan, re-references to the frequently-referenced working set incur misses.

Figure 2 illustrates the desired cache replacement behavior. When the working set is larger than the cache, the replacement policy should preserve *some* fraction of the working set in the cache so that at least *that* fraction gets cache hits. In the presence of recurring scans, the replacement policy should preserve the frequently-referenced working set in the cache. Addressing thrashing and mixed access patterns was important because they occurred very frequently in enterprise server applications and PC games, and in multiprocessor workloads with shared LLCs.

B. Casting Cache Replacement as a Prediction Problem

Rather than storing the LRU chain position in the n bits of the enhanced NRU scheme, we cast the cache replacement problem as a prediction problem. As such, we developed the Re-Reference Interval Prediction (RRIP) framework for improving cache replacement. RRIP uses the n bits to store the *Re-Reference Prediction Value (RRPV)* of a cacheline. With n bits per cacheline, there exist 2^n possible RRPVs.

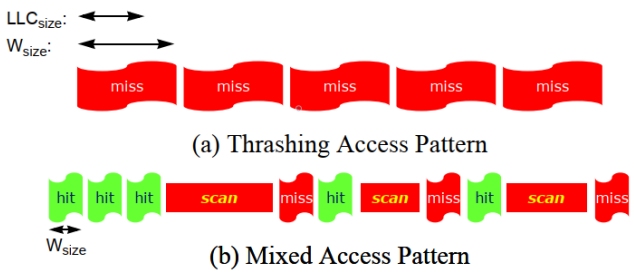

Fig. 1. The LRU Problem.

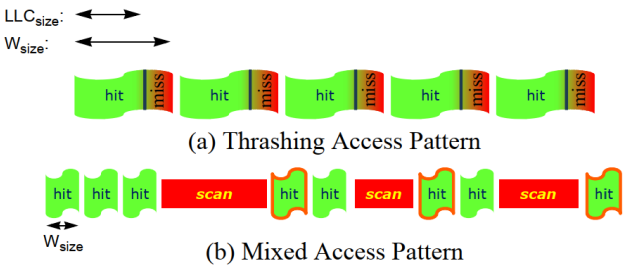

Fig. 2. The Desired Replacement Behavior.

A low RRPV is a prediction that the cacheline will likely be re-referenced soon, while a high RRPV speculates that the cacheline is more likely to be re-referenced only far in the future (if ever). The RRPV for a cacheline can be naturally assigned on cache insertions and updated on cache hits.

RRIP used the insight that new cachelines should not be given too much (or too little) time in the cache. If a new cacheline is given too much time, cache space is wasted when it is not used. If a new cacheline is given too little time, performance degrades due to early eviction. RRIP addressed this by statically selecting a suitable RRPV on cache insertion and *learn* the re-reference behavior of cachelines. Upon re-reference, RRPVs could be updated by either predicting that the line will be re-referenced soon (setting the RRPV to zero) or gradually reducing the RRPV based on access frequency. Using recency to update the RRPV was more scan resistant than using frequency. We referred to this policy as Static RRIP (SRRIP) and showed that it is scan-resistant for scan lengths that can be determined analytically. For cache associativity A, frequently-referenced working set size w ($w < A$), n-bit SRRIP is scan-resistant when:

$$\text{Scan}_{length} <= (\ 2^n - 1\ ) * (\ A - w\ )$$

C. The Need for Thrash Resistance

SRRIP provides scan resistance for mixed access patterns but behaves like LRU for thrashing access patterns (always incurs cache misses). In earlier work at ISCA 2007 [O25] by some of the authors and researchers at University of Texas at Austin, the LRU thrashing problem was solved using the Bimodal Insertion Policy which inserts the majority of cachelines at the "LRU position" [O25]. This enabled some of the working set to be preserved in the cache and receive cache hits. We extended this idea to RRIP by proposing thrash-resistant Bimodal RRIP (BRRIP). For $n = 2$, BRRIP inserts the majority of cachelines with an RRPV of $2^n - 1$ (i.e., 3) and some cachelines with an RRPV of $2^n - 2$ (i.e., 2).

D. A Quest for Both Scan Resistance and Thrash Resistance

However, we desired a cache management policy that is both scan resistant and thrash resistant. SRRIP provides scan resistance and BRRIP provides thrash resistance alone, but neither provides both. Thus, we needed a dynamic policy that adapts to the access pattern and uses SRRIP for mixed access patterns and BRRIP for thrashing access patterns. Our ISCA 2007 paper [O25] also proposed *Set Dueling* to allow a cache to select dynamically between two competing cache management policies. Using Set Dueling, we implemented Dynamic RRIP (DRRIP) that dynamically selects between SRRIP and BRRIP to improve overall cache performance. DRRIP is a low-overhead scan-resistant and thrash-resistant cache management policy. Using $n = 2$ provided a good storage/performance tradeoff. Figure 3 illustrates the RRIP state machine identifying both the SRRIP and the BRRIP component policies of DRRIP.

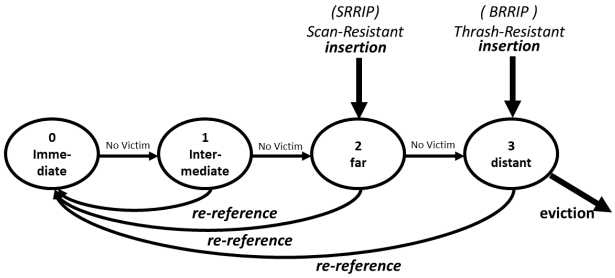

Fig. 3. RRIP State Machine.

III. Impact

Once the research and product groups understood they were taking complementary approaches toward the same goal, they combined their efforts, which enabled the development of a simple, practical cache management policy that ultimately outperformed LRU with less hardware than LRU. The replacement policy was originally created to help Haswell improve its performance over its predecessor, Ivy Bridge (IVB). However, IVB "stole Haswell's thunder" by adopting what we internally called "Quad-Age" (SRRIP) and "Adaptive Fill Policy" (DRRIP) [1] – helping boost IVB performance over its predecessor, Sandy Bridge. Shortly afterwards, academics reverse engineered the IVB replacement policy [3].

Though decades of research tried to address the problems of LRU, such proposals either incurred large storage overhead or required significant changes to the existing cache design. The beauty of RRIP was that it integrated seamlessly into the existing cache design. In fact, for $n = 1$, RRIP simply reverts to the existing NRU implementation. Furthermore, for $n > 1$, the RRPVs on hits and insertion could be configured to precisely mimic n=1. These properties made RRIP low risk and required minimal design, verification, and testing effort.

Our ISCA 2010 paper has also influenced cache management both in industry and academia. As of this writing, this paper has received more than 900 citations. RRIP provided the framework and cast the cache replacement problem as a prediction problem. This enabled future research to develop state-of-the-art policies (e.g., SHiP [4], HawkEye [2]) that improved prediction policies for where to insert an incoming line. Furthermore, there is undocumented evidence that RRIP is incorporated into cache designs and on-chip buffers of several microprocessors industry-wide.

References

[1] S. Jahagirdar, V. George, I. Sodhi, and R. Wells, "Power management of the third generation Intel Core micro architecture formerly codenamed Ivy Bridge," in *2012 IEEE Hot Chips 24 Symposium (HCS)*. IEEE, 2012, pp. 1–49.

[2] A. Jain and C. Lin, "Back to the future: Leveraging Belady's algorithm for improved cache replacement," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 78–89, 2016.

[3] H. Wong, "Intel Ivy Bridge cache replacement policy," Jan 2013. [Online]. Available: http://blog.stuffedcow.net/2013/01/ivb-cache-replacement/

[4] C.-J. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely Jr, and J. Emer, "SHiP: Signature-based hit predictor for high performance caching," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, 2011, pp. 430–441.