6.823 Computer System Architecture

SRRIP replacement policy

http://csg.csail.mit.edu/6.823/

The Least Recently Used (LRU) replacement policy selects the cache line that was accessed furthest in the past as a victim. LRU works reasonably well in practice, but suffers poor performance with some access patterns. Thus, modern processors use more refined policies.

A key problem of LRU is that it does not protect highly reused data from being displaced by data with poor reuse. For instance, consider a single set of a 2-way set associative cache that receives the following address sequence:

A, A, B, C, A, A, B, C, A, A, B, C…

Each letter represents a unique line address. Assume that all these line addresses are mapped to the same set of the cache. The optimal replacement policy would keep the line at address A in the cache since it has the minimum reuse distance, and always replace B or C. However, with LRU, accesses to B and C pollute the cache and displace line A, causing more misses.

SRRIP (Static Re-Reference Interval Prediction) [1] is a simple replacement policy that addresses this and similar pathologies. Recent Intel processors use variants of SRRIP in their last-level caches [2].

SRRIP prevents cache lines with a distant re-reference interval from evicting lines that have a near-immediate re-reference interval. SRRIP accomplishes this by tagging each cache line with an M-bit value, called the *Re-Reference Prediction Value* (RRPV). Intuitively, the *larger* the RRPV, the *longer* the line takes to be re-referenced, and hence the *less valuable* to keep it in the cache.

On a miss, when the line is inserted in the cache, its RRPV is set high (e.g., to $2^M - 2$). On each hit, the line's RRPV is set to 0. Thus, when a line is first inserted, SRRIP makes a pessimistic prediction by default, believing it will have a long re-reference interval. SRRIP starts predicting a short re-reference interval only after the line experiences a hit.

On a miss, SRRIP evicts the line with the highest RRPV. Because lines that receive a hit can become rarely accessed after a while, SRRIP needs a mechanism to de-prioritize old hits. SRRIP achieves this through a mechanism called *aging*: only lines with the maximum value of RRPV, i.e. ($2^M - 1$), can be selected as victims: if no line in the set satisfies this, the RRPVs of all lines in the set are incremented until there is at least one line with the maximum RRPV.

Finally, invalid lines always have the maximum value of RRPV, i.e., ($2^M - 1$), indicating that they have the highest priority to become victims.

The code below formalizes these rules, showing how SRRIP responds to an access in a set of a W-way set associative cache:

```
// Candidates and their RRPVs are indexed by w in [0, W - 1]

if access is a hit on way w:
    RRPVs[w] = 0

else: // access is a miss
    while (maximum value of RRPVs != 2^M - 1): // 3 if M == 2
        Increment all RRPVs by 1
    // Now there must be at least an RRPV with value 2^M - 1
    Select w as the minimum index s.t. RRPVs[w] == 2^M - 1
    // w-th candidate is selected as the victim
    Replace the old data of candidate w with the new data
    RRPVs[w] = 2^M – 2 // 2 if M == 2
```

The following state-transition diagram illustrates how the RRPVs are managed for 2-bit SRRIP, i.e., when using M=2-bit RRPVs, a common implementation choice.

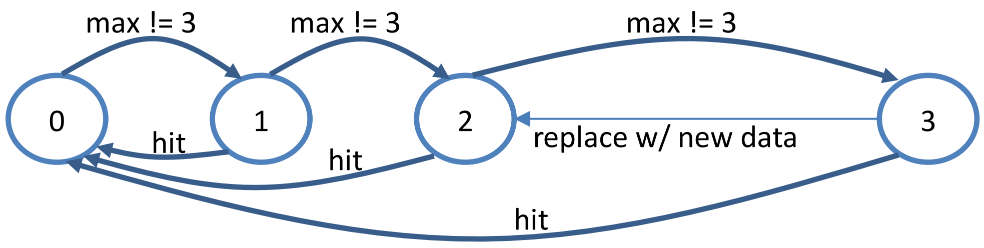

Figure 1. State-transition diagram of an RRPV in 2-bit SRRIP.

In 2-bit SRRIP, only lines with an RRPV of 3 can be evicted. The RRPV reaches 3 because the line either has invalid data or stays in the cache without being accessed for long enough.

Importantly, unlike LRU, SRRIP treats hits and first-time accesses differently. A hit sets the candidate's RRPV to 0, while a first-time access sets the candidate's RRPV to be ($2^M – 2$). Therefore, data that is accessed only once ages to the maximum RRPV quickly, and is evicted earlier than data that has experienced a hit.

This is precisely why SRRIP avoids the performance pathologies of LRU. Recall the example mentioned at the beginning. The initial high value in RRPV of line A is lowered to 0 in its subsequent hit, while the RRPVs of B and C never reach 0. Upon an eviction, this distinction in RRPVs prioritizes B and C to become victims, enabling a higher hit rate than in LRU.

References

[1] Jaleel, Aamer, et al., "High performance cache replacement using re-reference interval prediction (RRIP)," in *Proc. of the 37th Annual International Symposium on Computer Architecture*, 2010.

[2] Wong, Henry, "Intel Ivy Bridge Cache Replacement Policy," 2013. [Online]. Available: https://blog.stuffedcow.net/2013/01/ivb-cache-replacement/.