Filter Caching for Free: The Untapped Potential of the Store-Buffer

Spectredown (**# 4**)
Prajeeth.S
(190050117)

I. Summary

The authors have identified the potential of the store buffer to act as a filter cache. They have presented a unified store queue/buffer/cache to cache data that has been written back to memory and predict hits to avoid L1/TLB probes and save energy. They demonstrate increase in the Store buffer hit rates, reduction in dynamic energy on the SPEC2006 benchmark without causing avoiding performance degradation and incurring minimal storage overhead.

II. Details

- Standard Store-queue/buffer(SQ/SB) is a circular FIFO consisting of two components. Store queue(SQ) allows stores to retire under cache misses my maintaining order among stores. When the store is committed, it is moved from Store queue(SQ) to the Storage buffer(SB). It follows an aggressive eviction policy where the entry is SB is removed right after writing back to cache.
- Owing to the small size and aggressive eviction policy, SQ/SB utilization and the hit rate is low. So, programs are not able to benefit from the low latency of SQ/SB hits. SQ/SB intrinsically has the probe and copying overheads required for a filter cache.
- A portion of the unified SQ/SB structure is used as a Storage Buffer Cache(SBC), and the combined structure becomes S/QBC. To reduce accesses to L1/TLB, after writing back to cache from SB, the block is moved to the SBC, allowing maximum hit rates until there is a space requirement. In order to predict if the S/QBC gives a hit, a memory dependence predictor based on store-distances is used. Handling the SBC synonym problem(two Virtual addresses mapping to the same physical address) that occur requires at most one TLB access.
- To handle problems due to coherence(where another core can modify data unknown to S/QBC), the SBC entries are flushesd only under restricted conditions depending on the Cache invalidation protocol.
- To avoid flushing too many entries, the authors propose to add extra dirty bits to each cache line that correspond to a ”color” or ”epoch”. On Invalidation/Eviction at a cache line in L1, entries in SBC with the same color as the L1 line are flushed. The others remain unaffected.

III. Strengths

- The proposed SBC requires minimal storage overhead (only 2 or 3 bits per cache line).
- The SBC is a logical portion of the unified SQ/SB. So, moving entries from SB to SBC does not require any copying of data. It suffices to maintain a pointer to mark the boundary between SB and SBC and move it accordingly.
- Using only 2 extra bits(3 colors), the performance improvements nearly match that of the Optimal SQ/SB(which delays write back until there is a space requirement), and hence sufficient for practical usage.

IV. Weaknesses

- Degradation of performance is observed in cases with low read locality and inaccurate memory dependence predictors leading to serialization of S/QBC and L1/TLB probes.
- Performance Improvements are very minimal in case of Parallel Workloads as there are more frequent invalidations and evictions.
- Usage of a finite number of extra dirty bits for color does not eliminate unnecessary flushing from SBC caused due to a relatively old store.

V. Extensions

- The authors have described the coherence protocol for the Total Store Ordering memory model(TSO). This can be extended to weaker memory models.
- The coherence protocol has been described for the simple MESI cache invalidation protocol. This can be extended to more complex MOESI/MESIF protocols.