The Evicted-Address Filter: A Unified Mechanism to Address Both Cache Pollution and Thrashing


Vivek Seshadri


Onur Mutlu


Michael Kozuch


Todd Mowry

Off-chip main memory has long been a bottleneck for system performance. With increasing memory pressure due to multiple on-chip cores, effective cache utilization is important. In a system with limited cache space, we would ideally like to prevent 1) cache pollution, i.e., blocks with low reuse evicting blocks with high reuse from the cache, and 2) cache thrashing, i.e., blocks with high reuse evicting each other from the cache.

In this paper, we propose a new, simple mechanism to predict the reuse behavior of missed cache blocks in a manner that mitigates both pollution and thrashing. Our mechanism tracks the addresses of recently evicted blocks in a structure called the Evicted-Address Filter (EAF). Missed blocks whose addresses are present in the EAF are predicted to have high reuse and all other blocks are predicted to have low reuse. The key observation behind this prediction scheme is that if a block with high reuse is prematurely evicted from the cache, it will be accessed soon after eviction. We show that an EAF-implementation using a Bloom filter, which is cleared periodically, naturally mitigates the thrashing problem by ensuring that only a portion of a thrashing working set is retained in the cache, while incurring low storage cost and implementation complexity.

We compare our EAF-based mechanism to five state-of-the-art mechanisms that address cache pollution or thrashing, and show that it provides significant performance improvements for a wide variety of workloads and system configurations. Fig.1 shows the results for 4-core workloads.

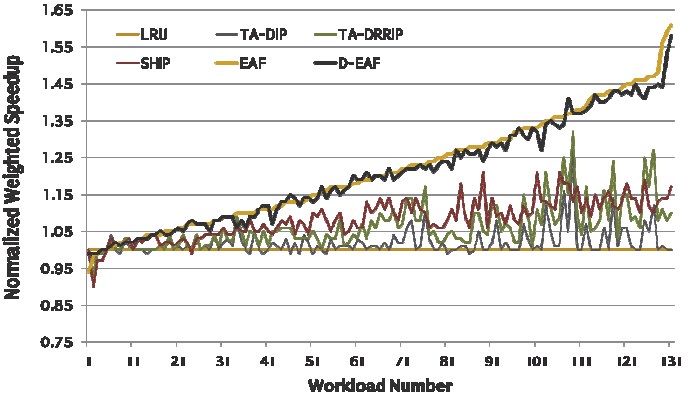

Fig. 1: 4-Core System Performance: EAF-Cache vs Prior Approaches.