PROPOSAL LIRS: Low Inter-reference Recency Set

Xien Thomas & Dalton Altstaetter
Texas A&M University

Summary

After a thorough review of cache replacement literature, we settled on selecting an algorithm called Low Inter-reference Recency Set (LIRS) by Jiang et al [1]. LIRS addresses both the limits of LRU and LFU by taking into account both recency (addressed by LRU) and the Inter-reference Recency (IRR) i.e. how many instructions are called in between the second reference to an instruction (addressed by LFU). Considering both these attributes prevents infrequent instructions from polluting the cache. In LIRS only cache entries with High Inter-reference Recency (HIR) are evicted and are placed in the HIR Set and viewed as “cold blocks”. Putting new entries in the HIR Set is what prevents cache pollution, whereas the LIR Set is for blocks that have been called recently and with higher frequency.

The following events characterize the hit/miss behavior:

- Cache Hit:
  - Check which block was hit.
  - If the block was in LIR Set
    - Update the block’s IRR by setting it equal to the stored recency value.
    - Update the recency of the block to 0 indicating its the most recent instruction called.
    - Increment all other cache entry’s recency by 1.
    - Note: No eviction is done and all entries remain in the cache
  - If the block was in the HIR Set
    - Update the block’s IRR by setting it equal to the stored recency value.
    - Update the recency of the block to 0 indicating its the most recent instruction called.
    - Compare the IRR of the current block to the blocks in the LIR set
      - If current_IIR < max(IRR_of_all_entries_in_LIR_Set)
      - Remove the cache entry in LIR Set with the IRR_max value and place it in the HIR Set.
    - Increment all other cache entries recency by 1.

- Cache Miss:
  - Evict the HIR entry and place the current block in the HIR Set
  - Update the current block’s IRR (if applicable)
  - Increment all other cache entry’s recency by 1.

Discussion

We chose this approach because we were interested of how much it differs from the LRU and LFU approaches. LIRS tells the frequency of each candidate block and how recent the blocks were. The approach is another way of combining LRU and LFU to get the benefits of both recency and frequency. We expect that the MPKI to be lower than both LFU and LRU, but can only hypothesize when compared to SRRIP. What we would like to investigate is how LIRS will do compared to SRRIP since both algorithms take different approaches for solving the downsides of LRU and LFU. We hope to identify any general instruction access patterns for which to compare LIRS vs SRRIP performance and conclude which cache replacement policy performs best based on MPKI, cycles, and CPI.

Outline

- Week of Nov. 12th
  - Setup environment
  - Begin implementing the LIR algorithm in Zsim
- Week of Nov. 19th
  - Finish any debugging and validate the algorithm via simulation
  - Run the benchmarks for LIR
- Week of Nov. 26th
  - Begin analysis of LIR against LFU, LRU, and SRRIP
- Final Week
  - Finish all analyses and submit the final report

References

[1] S. Jiang and X. Zhang, "LIRS: An efficient low inter-reference recency set replacement policy to improve buffer cache performance," in Proc. ACM SIGMETRICS Conf., 2002.