Computer Architecture (CS422) Review

Critical Paper Review: High-Performance Cache Replacement Using RRIP
Reviewer: Manish (190477)

Summary
Optimal policy(Belady) knows the re-reference interval but Practical cache replacement policies try to predict the re-reference interval. LRU always predict near-immediate re-reference which is problematic in the case of both scans(burst of reference to non-temporal data) and thrashing. In contrast, MRU always predicts distant re-reference which is problematic in the case of data locality. Dynamically changing between both using set dueling can be helpful in thrashing but have more hardware overheads and doesn't provide scan resistance. SRRIP predict intermediate re-reference interval on miss and is scan resistant, one variant of SRRIP named SRRIP-HP(SRRIP-Hit Priority) on hit predicts near immediate re-reference while another variant named SRRIP-FP(Frequency Priority) predicts that frequently referenced will have near immediate re-reference. SRRIP-HP performs significantly better than SRRIP-FP. SRRIP-HP outperforms LRU by 4% on single-core and 7% on multi-core. DRRIP uses switches between SRRIP-HP and BRRIP(Bimodal RRIP) based on the application using set dueling. DRRIP is both scan and thrash-resistant. DRRIP outperforms LRU by 10% on single-core and 9% on multi-core.

Details
Challenges to Cache replacement Policies:

- Shouldn't change existing cache structure and should have fewer hardware overheads.
- Should be resistant to scans(burst of reference to non-temporal data) and thrashing.
- Should perform well on all applications and be easy to implement.

How DRRIP solve the above challenges:

- Can be easily implemented on existing cache structure by only adding 2 bits per cache line to store the Re-reference Interval Value(RRIV).
- It has fewer hardware overheads than the existing policies (2x less than LRU and 2.5x less than LFU) on a 16-way set associative cache.
- It is resistant to both scans and thrashing. It dynamically chooses between SRRIP-HP and BRRIP using set dueling based on the application which is running.
- SRRIP-HP inserts new line with RRIV=2(long re-reference interval). On hit, it changes RRIV to 0(near-immediate re-reference interval). SRRIP-HP gives chance to improve and learn re-reference prediction by giving RRIV=2 to the inserted block instead of 3. SRRIP-HP also became scan resistant by giving RRIV=2 instead of giving 0. It is not thrash-resistant.
- BRRIP component of DRRIP helps in case of thrashing. It inserts the majority block with RRIV=3 and a few with RRIV=2. This helps in retaining some blocks in case of thrashing.
- Victim selection policy select block with max RRIV.

Evaluation and Results:

- It is evaluated on 14 benchmarks from 4 categories(PC games, Multimedia, Server, SPEC CPU2006). These are memory latency-sensitive tasks. All performance gains are reported on these benchmarks. On non-memory-intensive tasks, there are no such improvements.
- Tested both on single-core and multi-core(4 cores). All possible combinations of multi-core processes for 14 benchmarks (1001) are tested.
- Value of bits needed per cache line is calculated empirically based on the results obtained by running benchmarks. The optimal value is 2.
- SRRIP-HP and SRRIP-FP both tested on these benchmarks. SRRIP-HP performed better, so chosen in DRRIP.

Other Details:
Tradeoff: Best performance got on 3 bits per cache line but chooses 2 bits per cache line as 3 bit require more hardware for which performance gain is not significant.
DRRIP cannot give performance gains on L1 and L2.

Positives

- Uses significant less hardware for 16-way associative cache with performance gains over LRU, LFU etc. It also doesn't change the existing cache structure which makes it more practical to use and easy to implement over other policies.
- Evaluation mechanism for Multi-core experiments is thorough and robust as it considers all possibilities of $\binom{14}{4}$ over 4-core. Benchmarks are chosen over a wide range of applications rather than selecting a single benchmark like SPEC CPU which doesn't properly represent the complete range of applications.
- Well thought pre-analysis of all 14 benchmarks over the range of memory sizes which helps in analyzing the results and predicting will this policy helps on specific workload with given cache size. A wide range of variables is considered and experimented with like insertion value, no. of bits per cache line, size of the cache, associativity, single-core vs multi-core, different cache levels(2 and 3) and different hit policies.

Negatives

- Associativity experiments results are not shown, just claimed that get performance gains for all associativity. How performance is effected with change in associativity and No. of bits per cache line can dictate the policy we want to use for our cache.
- When comparing to modified LRU which insert at the specific position based on the application using Set Dueling author questioned the scalability when associativity of cache increases because the number of monitor sets can exceed the total number of sets. But in DRRIP they also use set dueling which can also face the same issue.

- Doesn't get performance benefit on every cache level. It works best on LLC but does not gain any performance on L1 or L2 due to more data locality. Got performance benefit on 512kb LLC. For some systems 512kb can be L2 size and LLC can even bigger. Author didn't experiment with changing L2 size to see benefits.
- Claimed 2x and 2.5x less hardware overhead than LRU and LFU respectively. But only mentioned in the appendix section that it is for a 16-way cache. For another associativity, it may decrease like for 8-way cache, hardware overhead is only 1.5x less than LRU.

Possible Extensions

- As we got performance benefits on some workloads and on some workloads DRRIP performed similarly to others which can also have fewer hardware overheads. What we can do is choose a lot of different kinds of benchmarks and analyze performance on them. From that analysis, we can decide to which industry this policy can be more beneficial rather than using it for every LLC.
- We can also experiment with how different applications, cache sizes and associativity affect the number of bits required per cache line, which can further reduce hardware overheads or improve performance using more hardware specific to certain applications which are more critical to us. Also how changing system configurations like L2=512kb and L2 replacement policy to DRRIP effect performance.