CS422 Project Report

Rohith Mukku (14402), Bhangale Pratik Anil(14173)

April 2017

Abstract

The development of caches and caching is one of the most significant events in the history of computing. A cache's eviction policy tries to predict which entries are most likely to be used again in the near future, thereby maximizing the hit ratio. The commonly used LRU replacement policy always predicts a near-immediate re-reference interval on cache hits and misses. Applications that exhibit a distant re-reference interval perform badly under LRU. Modern caches extend the usage history to include the recent past and give preference to entries based on recency and frequency. An alternative replacement algorithm using Static Re-reference Interval Prediction (RRIP) is based on this fact. Static RRIP (SRRIP) requires additional n-bits per cache block for prediction and can be easily integrated into existing LRU approximations found in modern processors. We further tried improving SRRIP with Signature-based Hit Predictor(SHiP) algorithm on top of original SRRIP. SHiP tries to predict re-reference interval for each cache block using Program Counter(PC) as a signature. This report compares SRRIP and SHiP with LRU using SPEC CPU2006 workloads with a 2MB last-level cache (LLC). Intel PIN tool's CACHE simulation API is used for the analysis of LLC.

Introduction

Today's cache replacement policies make their replacement decisions depending on predictions of which block will be re-referenced farthest in the future and pick that block for replacement. On a miss, replacement policies make a prediction on when the missing block will be re-referenced next.

As described above, LRU replacement predicts that a block filled into the cache has a near-immediate re-reference interval, thus, clears out the distant block. Whereas, SRRIP makes its replacement decision based on the history or frequency of block accesses.

Re-Reference Interval Prediction (RRIP) chain is a chain that represents the order in which blocks are predicted to be re-referenced. The block at the head of the RRIP chain is predicted to have a near-immediate re-reference interval while the block at the tail of the RRIP chain is predicted to have a distant re-reference interval. A near-immediate re-reference interval implies that a cache block will be re-referenced in the near future while a distant re-reference interval implies that a cache block will be re-referenced in the distant future. On a cache miss, the block at the tail of the RRIP chain (i.e., the block predicted to be referenced most far into the future) will be replaced. LRU replaces the most distant one, i.e., the block which is not used for long time. On the other hand, SRRIP assigns each cache block re-reference Prediction Values (RRPV) which will be used in prediction. RRPV for near-immediate re-reference interval will be close to 0 and for distant, RRPV will be maximum.

SRRIP can be implemented in two ways:-

1. SRRIP-HP: update the re-reference prediction with Hit Priority (HP)
2. SRRIP-FP: update the re-reference prediction with Frequency Priority (FP)

SHiP which is an improvement over SRRIP algorithm tries to predict the re-reference interval for every cache block. For this purpose we would need to add some extra attributes along with cache line. We add integer to keep track of signature of current block and bit to check if it has been hit after brought into the cache. Signature of the block is used to store identity of the block. Different signatures can be memory region signatures (SHiP-Mem), program counter signatures (SHiP-PC), and instruction sequence history signatures (SHiP-ISeq). For our analysis we will use Program counter as a signature of a cache block. We will also need a global data structure(SHCT) to store the data of cache block to predicts its re-reference interval and we will simply use modulo hash to index the SHCT. When the application working set is larger than the available cache or when the application has a mixed access pattern where some memory references have a near-immediate re-reference interval while others have a distant re-reference interval. In LRU and SRRIP we had a constant re-reference interval, hence in such cases we had a higher miss rate in LLC. Using SHiP we would dynamically set the re-reference interval.

Design

The design implemented for this project:

- Cache has a three-level hierarchy with L1 instruction cache being a 32 KB, 8-way 64-byte block size, LRU and L1 data cache is a 32 KB 8-way 64-byte block size, LRU.
- L2 cache is a 256 KB 8-way 64-byte block size LRU. L3 cache is a 2 MB 16-way 64-byte block size.
- L1, L2 follow LRU replacement policy.
- SRRIP implemented was n-bit SRRIP with n=4.

SRRIP (n-bit, modified) Policy:

- Each block's RRPV was initialized to (2^n-1), the distant re-reference interval.
- Every time, if a block is placed/replaced in the cache, the block is assigned RRPV value to (2^{n-1}). In case of original SRRIP, every time a block is brought into the cache, its RRPV value is set to $(n-1)$, but in experiments we found that in most cases the newly brought cache block gets replaced and hence resulting in fewer hit rate. Hence to balance LRU and SRRIP, we would st RRPV to (2^{n-1}).
- For SRRIP-HP: Every time a cache hit happens, the block's RRPV value is decremented to 0.
- For SRRIP-FP: Every time a cache hit happens, the block's RRPV value is halved.
- For a cache miss, the first block with the highest RRPV value gets replaced.

SHiP Policy:

- In addition to SRRIP, SHCT of 5000 size is kept globally, to keep track of behaviour of cache block.
- Every time cache block hits, we change hit bit to 1 and also increment SHCT corresponding to that signature.
- If cache block misses, we find a replacement candidate exactly like in the case of SRRIP. If the replacement candidate has its hit bit set to true, in such cases we decrement SHCT for corresponding signature (distant re-reference).
- For the new cache block in LLC, we assign its signature as PC modulo 5000 and also set its hit bit false. Now we look in SHCT for current signature if it is 0, then we predict that this block will not have access in near future and hence set its RRPV to $(3 * n/4)$ else we will assume this block has access is near future and hence set its RRPV to $(n/4)$.

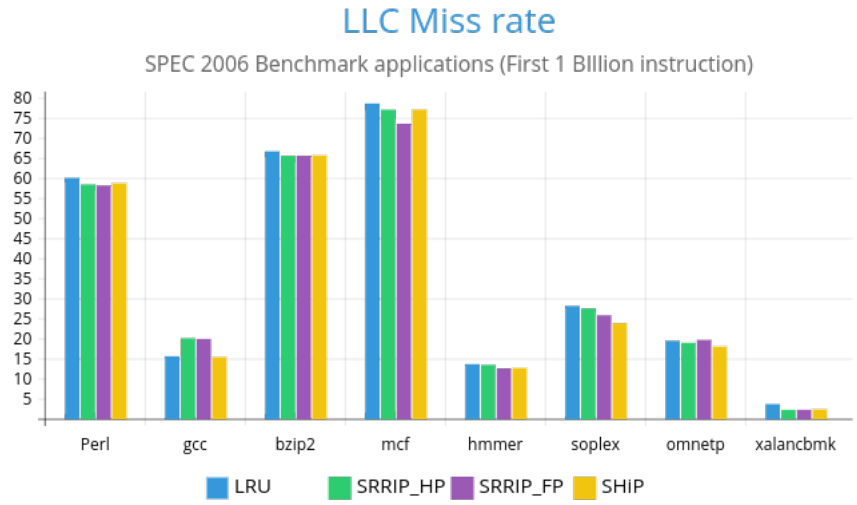
Observations & Results

Cache Statistics

	perl	gcc	bzip2	mcf	hmmer	soplex	omnet	xalanc
LRU Cache Hits	673899	20964867	4823678	3260483	2119618	885856	478961	5887966
LRU Cache Misses	1010496	3858460	9626345	11933611	332968	345792	115286	221563
SRRIP-HP Cache Hits	696098	19802633	4988243	4022358	2139940	893043	490476	6051271
SRRIP-HP Cache Misses	976362	4963090	9462485	11170078	330765	337939	80852	136182
SRRIP-FP Cache Hits	703223	19842236	4984603	4024052	2139973	895624	473027	6015674
SRRIP-FP Cache Misses	975017	4925362	9461712	11169044	305714	310468	114087	136219
SHiP Cache Hits	689694	20999888	4969868	3492522	2149467	919383	522203	6076651
SHiP Cache Misses	978610	3798992	4969868	11698328	309916	286978	114345	138696

Cache Miss Rates

	perl	gcc	bzip2	mcf	hmmer	soplex	omnet	xalanc
LRU Cache Miss Rate	59.99	15.54	66.61	78.54	13.57	28.07	19.4	3.62
SRRIP-HP Cache Miss Rate	58.37	20.04	65.48	77.00	13.38	27.45	18.91	2.20
SRRIP-FP Cache Miss Rate	58.09	19.88	65.49	73.51	12.50	25.74	19.43	2.21
SHiP Cache Miss Rate	58.65	15.31	65.60	77.00	12.60	23.78	17.96	2.31



Conclusion

In most of the cases, SRRIP has lesser miss rate (more hit rate) as compared to LRU. Also SHiP has lesser miss rate (more hit rate) as compared to SRRIP. Using more complex algorithm to predict the re-reference interval, we can achieve higher hit rate. Also in our case, finite number of instructions are executed to produce results, executing more number of instruction using fast forwarding will definitely improve statistics of hit rate.

References

- Cache Replacement Policies
  https://en.wikipedia.org/wiki/Cache_replacement_policies
- High Performance Cache Replacement Using Re-Reference Interval Prediction (RRIP)
  http://people.csail.mit.edu/emer/papers/2010.06.isca.rrip.pdf
- SHiP: Signature-based Hit Predictor for High Performance Caching
  http://mrmgroup.cs.princeton.edu/papers/MICRO11_SHiP_Wu_Final.pdf
- Intel Pintool
  https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool