Accuracy of Different Cache Replacement Implementations

A. Emis, *Student, UCLA*

I. LRU and SSRIP Base Implementation

I was given the Least Recently Used Insertion Policy and Static Re-Reference Interval Prediction base implementations. The goal of this project was to implement different cache replacement algorithms to minimize the ratio of misses to total cache accesses. Using LRU and SRRIP means evicting the cache entry that has been least recently used when the cache is full, to make room for the newest cache entry. When inserting the newest cache entry, it may seem most logical to insert it at the Most Recently Used position, since it was accessed most recently. However, this implementation installs the cache entry at LRU and only promotes it to MRU if it is hit. This makes sense because maybe that address is only accessed one time, so keeping it in the cache for a long time, while other more important cache entries are evicted, can cause wasted space in a cache, where space is already very limited. With the implementations I was given, the LRU algorithm inserted the cache entry with a value of 0 every time. The SRRIP algorithm inserted the cache entry with a value of 15 originally and promoted it to a value of 0 if it was hit. The LRU algorithm yielded a cache miss rate of 0.5118 while the SRRIP algorithm yielded a cache miss of 0.6925. I suspected the SRRIP performed so poorly compared to LRU because the value was 15, which meant there was a high likelihood of that cache entry getting evicted during the next cycle. In other words, it didn't have much of an opportunity to be hit again. I explore refining these parameters in section III of this paper.

II. Bimodal Insertion Policy

After doing some research on some other cache replacement policies, using [1], [2], and [3] as resources, I decided to use Bimodal Insertion Policy. Instead of always inserting at the LRU position, the program infrequently inserts lines in the MRU position. This is done by using a random number generator where if the outcome is less than a certain variable, called the bimodal throttle parameter, then that cache entry is inserted at MRU. This means that there is a chance of a useful cache being inserted at MRU instead of LRU, where it would probably be evicted before it is hit. I tried to find documentation on the ideal bimodal throttle parameter value, but unfortunately had to resort to guessing and seeing which values were the most accurate. I ended up generating a random integer and checking if it was a multiple of 2048 by using the % (mod) operator. If it was, this cache entry would be inserted at the MRU position. This technique yielded a cache miss rate of 0.5036.

III. Refining Parameters

Refining parameters ended up being much more important that I had originally thought. I had to find the most ideal values for the LRU insertion parameter and the bimodal throttle parameter. The MRU insertion parameter value was straight forward because that just required setting the value to 0, implying it was most recently used. Below are the tables to describe the LRU values and bimodal throttle parameter values I tried and their respective cache miss rates.

LRU Value	Cache Miss Rate
4	0.5090
7	0.5057
8	0.5054
9	0.5077
10	0.5161
14	0.6602
15	0.6925

Bimodal Throttle Parameter Value (LRU Value = 8)	Cache Miss Rate
32	0.5053
64	0.5049
128	0.5046
256	0.5042
512	0.5039
1024	0.5038
2048	0.5036
4096	0.5036

IV. Other Attempts

I wanted to briefly discuss the other attempts I made with this project. My main idea was to dynamically switch between the LRU and BIP techniques. I would do this by using a counter that would increment/decrement if one technique had a hit, while the other had a miss. If the counter was 0 or 1, LRU would be used. If the counter was 2 or 3, BIP would be used. Unfortunately, I wasn't able to figure out how to check if there was a hit for both techniques. I could only know the hit or miss for one technique. I attempted to still use a counter in a variety of different ways, but it wasn't as accurate as using just BIP.

V. Conclusion

After some experimentation, as seen on the tables above, the ideal MRU value was 0, the ideal LRU value was 8, and the ideal bimodal throttle parameter value was 2048. When I used BIP with 2048, I got a lowest cache miss rate of 0.5036. With my implementation, there is a big table with all the ints determining if the cache was recently used or not, which is a storage overhead that might be needed to take into consideration if there is a storage limitation.

References

1. G. Lee, B.J. Kim, E. Chung. *"Exploring Replacement Policy for Memcached."* https://dtl.yonsei.ac.kr/docs/International_Conference/[2020][ISOCC]Exploring%20Replacement%20Policy%20for%20Memcached.pdf
2. A. Gupta. *"Adaptive Insertion Policies for High-Performance Caching."* 13 February 2019. https://medium.com/@arpitguptarag/adaptive-insertion-policies-for-high-performance-caching-a741c52f515c
3. O. Mutlu. *"Lecture 3: Cache Management and Memory Parallelism."* https://safari.ethz.ch/architecture/fall2017/lib/exe/fetch.php?media=onur-comparch-fall2017-lecture3-afterlecture.pdf