Proposed Enhancements to Fixed Segmented LRU Cache Replacement Policy

Kathlene Hurt and Byeong Kil Lee
Department of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio, United States
kathlene.hurt@gmail.com

***Abstract*—The goal of this research is to design a low cost cache replacement algorithm that achieves comparable performance to existing scheme. In previous work, we proposed an enhancement to the SLRU (Segmented LRU) algorithm called fixed SLRU that fixes the number of protected and probationary segments. Due to an ineffective simulation environment, we were unable to fully understand our results. In this work, we will test fixed SLRU with an improved simulation environment, and propose three enhancements to the algorithm: cache bypassing, random promotion, and aging. Our experiment results show that fixed SLRU achieved an average speed up of 7.00% over LRU, and 4.22% over SLRU, with a maximum speed up of 43.55% over LRU, and 41.59% over standard SLRU. With certain enhancements applied, we achieved a maximum speed up of 61.70% over LRU.**

Keywords-cache, replacement policy, segmented, LRU.

I. INTRODUCTION

The basic idea of the SLRU replacement policy is that if a line has been accessed while occupying the cache, it should be more difficult to evict than a line that has never been accessed. This scheme is implemented by dividing the cache set into two segments: *the protected segment* and *the probationary segment*. All lines entering the cache are initially part of the probationary segments. If a cache hit occurs on a line in the probationary segment, that line is then promoted to the protected segment. All victims are selected from the probationary segment using LRU replacement policy. Cache lines in the protected segment are only victimized if the probationary segment is empty. The advantage of SLRU over LRU is that it better exploits the temporal locality of lines by protecting more frequently used lines.

Based on our observation with the SLRU algorithm in previous work, we found that often, only one or two lines occupied the protected segment. We proposed a SLRU algorithm with a constant number of protected and probationary ways, called *fixed SLRU* [3]. In this work, we will further explore the performance gain of fixed SLRU over SLRU, and propose three enhancements: cache bypassing, random promotion, and aging.

We use ten SPEC2K6 benchmarks in the development, and verification of this algorithm. All simulations are executed using the default cache configuration laid out by [2]: a 1K 16-way set associative cache, with 64 bit block size. The ten benchmarks chosen are considered "cache sensitive". This was determined by achieving at least a 1% performance improvement by increasing the LLC cache size from1M to 2M using the true LRU policy[1].

II. SLRU WITH FIXED SEGMENT SIZE

To describe the fixed SLRU policy, we use the notation N:P, where N is the number of protected segments, and P is the number of probationary segments. The ratio between protected and probationary segments affects the cache performance. In order to find the optimum ratio, multiple simulations were executed. Figure 1 shows the results of these simulations.

As shown in Figure 1, five benchmarks experienced their greatest speedup over LRU while using the 11:5 segment ratio. Two benchmarks, gobmk, and mcf, experienced their greatest CPI at 15:1, and two benchmarks, zeusmp, and gromacs , experienced their greatest CPI at 1:15. Using this information, we determined that the 11:5 ratio would be the ideal segment ratio to use in our algorithm, which is consistent with previous work [3]. When using the 11:5, no benchmark experienced a decrease greater than 2.9% in CPI compared to its ideal segment ratio.

Figure 2 shows the CPI speed up over LRU and SLRU when the fixed SLRU algorithm was applied using the 11:5 segment ratio. A 7.00% average speed up was achieved over LRU, and 4.22% over SLRU. We had a maximum speed up of 43.55% over LRU, and 41.59% over SLRU. Over LRU, only one benchmark, soplex, experienced a decrease in CPI (2.7%). Over SLRU, two benchmarks experienced a

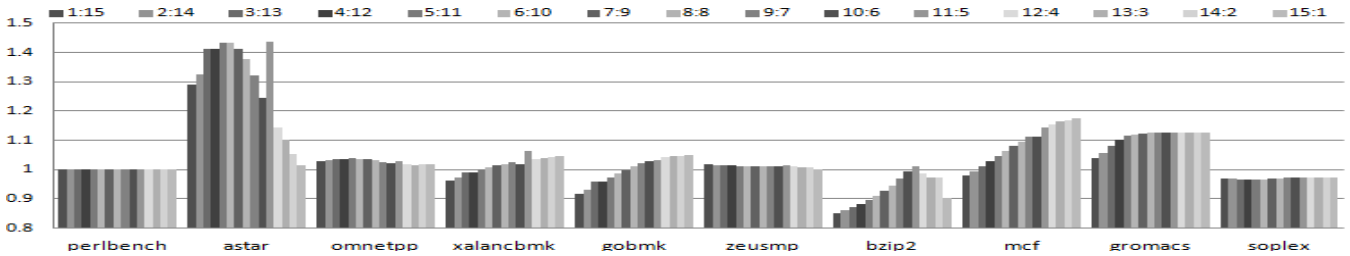

Figure 1: Effect of Segment Ratio on CPI Speed Up over LRU

978-1-4799-3214-6/13/$31.00 ©2013 IEEE

decrease in CPI, but neither was greater than 2.97%. When using the ideal segment ratio for every benchmark, an average speed up of 7.54% over LRU, and 4.73% over SLRU was achieved.

III. ENHANCEMENTS

Our first enhancement to fixed SLRU is a cache bypassing method which selects instructions to be bypassed based on their reference history. We predict that blocks that were not referenced when they last occupied the cache will tend to be dead the next time they enter the cache. To implement this, an additional one bit per cache line is required to represent whether a line has been accessed at least once while in the cache. If it has not, its tag will be updated to a table, called the bypass table. Based on our experiments, we found the bypass table to be most effective when it held a maximum of 64 tags. After a tag has been used to bypass the cache once, the line is cleared from the table. Tags were first written to any empty lines in the table. If no empty lines are available, tags are overwritten to a random line in the table.

An average speed up of 7.15% over LRU, and 5.11% over SLRU was achieved, with a maximum speed up of 48.97%, and 46.94%, respectively. This is 0.15% higher over LRU, and 0.89% higher over SLRU than fixed SLRU alone. Of the ten benchmarks tested, six benchmarks experienced a slight decrease in CPI (up to 3.01%) compared to 11:5 fixed SLRU. Since so many benchmarks experienced a decrease in CPI, and the overall speed up was negligible, we concluded that cache bypassing is not an effective enhancement to fixed SLRU.

It has been shown that randomly promoting lines from the probationary segment to the protected segment can increase performance in SLRU [1]. We propose an enhancement that randomly promotes lines to the protected segment as they enter the cache. After some experimentation, we concluded that random promotion would not be a beneficial enhancement to fixed SLRU. We found that even at very small probabilities (0.001%), nine benchmarks experienced a decrease in CPI. The benchmark astar was the only benchmark shown to benefit from random promotion. Astar was able to achieve a maximum speed up of 61.7% over LRU, which is a 41.8% increase over fixed SLRU alone, using a 20% random promotion probability.

Aging is another common enhancement to SLRU [1]. In standard SLRU, aging is the ability to remove lines from the protected segment, and return them to the probationary segment. Since the segment sizes in standard SLRU are dynamic, removing a line from protected segment does not affect the lines in the probationary segment. This is not the case for fixed SLRU. In fixed SLRU, removing a line from the protected segment means promoting a probationary line to keep the segment ratio consistent.

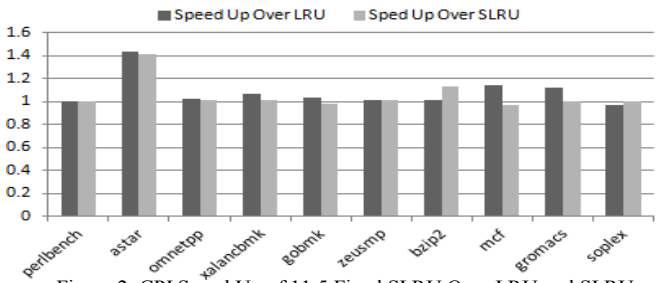

Figure 2: CPI Speed Up of 11:5 Fixed SLRU Over LRU and SLRU

To implement the idea of aging in fixed SLRU, we decided to alter the way the algorithm handles evictions from the protected segment. Currently, when a line is promoted to the protected segment by a cache hit in the probationary segment, the LRU line in the protected segment is demoted to the probationary segment, and it is set to be evicted on the next cache miss. To implement aging, we will instead maintain this lines stack position. This means that the line will not necessarily be evicted on the next cache miss, and gives it an increased chance to be promoted back into the protected segment.

We achieved an average speed up over LRU of 4.78%, and an average speed up over SLRU of 2.14%, with maximum speed ups of 17.10% and 15.51%, respectively. Only one benchmark, bzip2, experienced a performance boost from aging (11.66% CPI speed up over fixed SLRU). This is an extremely large decrease in CPI compared to 11:5 fixed SLRU alone. For this reason, we concluded that aging is not an effective enhancement to add to fixed SLRU.

IV. ANALYSIS OF RESULTS

The only enhancement to fixed SLRU that showed an overall increase in CPI was cache bypassing, but too many benchmarks experienced a decrease in CPI for cache bypassing to be considered an effective enhancement. Random promotion, and aging were shown to decrease performance when applied to fixed SLRU. Overall, we concluded that none of the enhancements added to fixed SLRU were effective.

V. CONCLUSION

Fixed SLRU using the 11:5 segment ratio achieved a 7.00% average speed up over LRU, and 4.22% over SLRU. We had a maximum speed up of 43.55% over LRU, and 41.59% over SLRU. This is significant considering fixed SLRU does not require any additional hardware over SLRU, and only 1 extra bit over LRU. None of the enhancements explored in this work were beneficial to fixed SLRU.

ACKNOWLEDGEMENTS

This research was supported by National Science Foundation under Grant No. CNS 1063106.

REFERENCES

[1] H. Gao, and C. Wilkerson. "A dueling segmented LRU replacement algorithm with adaptive bypassing," 1st JILP: Cache Replacement Championship, France, 2010.

[2] 1st JILP Workshop on Computer Architecture Competitions: Cache Replacement Championship. jilp.org, 2010 [July 2012].

[3] K. Morales, and B. Lee. "Fixed Segmented LRU Cache Replacement Scheme with Selective Caching, " 31st IPCCC, Dec. 2012.