Prediction Oriented Analysis of Optimal Replacement

Liu Fang [#1], Zhang Shengbing [*2], Ren Meng [#3], Zhang Meng [*4]

[#] *School of Computer Science and Engineering, Northwestern Polytechnical University Xi'an China 710072*

[1]xueshengliufang@126.com

[3]975290589@qq.com

[*] *School of Computer Science and Engineering, Northwestern Polytechnical University Xi'an China 710072*

[2]Zhangsb@nwpu.edu.cn

[4]zhangm@nwpu.edu.cn

Abstract—**The optimization of memory latency is always an important bottleneck to improving the performance of computer systems. The memory system, especially the last-level cache (LLC) as the important method to solve the "Memory Wall" problem, its management has become a key factors of influencing the performance of processor. And prefetching technology can improve the overall performance of the system by reducing pipeline stalls according to the temporal and spatial locality.**

This article is based on the characteristics of different workloads to study the performance of state-of-art LLC management policies with prediction technology. We achieve Bimodal Insertion Policy (BIP) which can adapt to changes in the working set. In order to further reduce the cache miss rate, we use the Set Dueling mechanism to dynamically choose the best replacement policy between Static Re-Reference Interval Policy (SRRIP) and Bimodal Re-Reference Interval Policy (BRRIP) based on the historical information [13]. We take SPLASH-2 as the benchmark to test the performance of these replacement policies. Finally we give a summary on the characteristics of different kinds of policies.

***Keywords*: LLC; replacement policy; CRC**

I. INTRODUCTION

In the Multi-core processor, Least Recently Used (LRU), the classical cache replacement, gradually loses its performance due to the cache dead block problem caused by high cache associativity. And there are many workloads that have a working set greater than the available cache size, to improve the performance of those workloads, dynamic insertion policy, which can effectively solve such problems of LRU, has received much more attention.

When the object cache is full, the policy decides which line is evicted for storing an incoming line. Usually there are two important positions: LRU position and Most Recently Used (MRU) position. We achieve the BIP which places most of incoming lines in the LRU position, and others in the MRU position with a low probability. BIP adapts to changes in the working set while retaining the thrashing protection of LRU policy. Then we use Set Dueling mechanism to dynamically choose the best policy between LRU policy and BIP to be applied to an incoming line [1]. In order to further improve the performance of system, we achieve 2 bits Dynamic Re-Reference Interval Policy (DRRIP) which uses Set Dueling to choose between its two component policies: SRRIP and BRRIP. DRRIP uses a counter per cache line to store the lines re-reference prediction value (RRPV) [13].

In order to compare different cache replacement algorithms, we use the Cache Replacement Competition (CRC) which can provide the competitors with a trace driven simulation environment [12]. Due to the long latency of the LLC and the low frequency of LLC accesses, algorithm complexity was not limited. Our evaluation, with typical benchmarks of SPLASH-2, shows that DIP can improve performance by 12.64% on average, and DRRIP can improve performance by 11.57% on average.

II. INSERTION POLICY

A. Analysis of DIP

The traditional LRU replacement policy always discards the least recently used item and inserts the coming line in the MRU position, that may retain the lines in the non-LRU positions of the recently stack even if they cause to contribute to cache hits, thus each time a cache line is used, the age of all other cache lines changes. BIP can improve the situation by placing the incoming lines in the MRU position (with low probability). BIP has a parameter ε which controls the percentage of incoming lines that are placed in the MRU position [1]. We set ε = 1/32, namely inserting incoming lines to LRU position 31 times and only one time to MRU position.

The compromise is dynamically choose between traditional LRU replacement policy and BIP depending on which policy incurs fewer misses, taking into account the workloads difference. We use the Set Dueling mechanism to implement DIP without requiring significant hardware overhead [1]. Assuming there are N sets in caches, each set concludes several cache lines, and K denotes the number of sets applied each policy ($K=2^N$). Thus we divide cache into K area with equal size which is N/K. The storage overhead we need is only 2 bits in order to detect what kind of replacement policy each set are needed. For a cache with N sets, there are $\log_2 N$

bits and we use $\log_2 K$ bits to identify the constituency and the remaining $\log_2 N/K$ bits to identify the offset from the first set in the constituency in Reference [1].

For all experiments in this section the value of N is 16, namely a cache containing 16 sets, and K is 4. We use 2 bits to identify the sets applied LRU or BIP, set 0 and every set 5rd set to LRU, and set 3 and every 3rd set to BIP. Auxiliary Tag Directory (ATD) has the same associativity as Main Tag Directory (MTD) of cache. We named a saturating counter as Policy Selector (PS), PS can keep track of which ATD-LRU and ATD-BIP incurs fewer misses. If there are a miss in ATD-LRU increases PS and a miss in ATD-BIP decreases PS [1]. The Most Significant Bit (MSB) of PS is an indicator of which of the two policies incurs fewer misses. The schematic of DIP is shown in Figure 1.

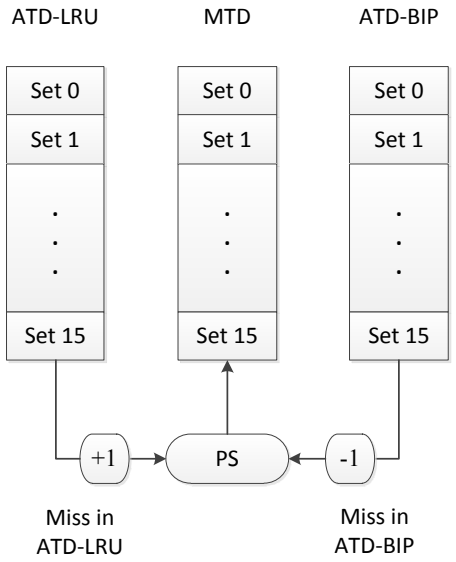

Figure 1: The Schema of DIP [1]

From Set 0 to Set 15 in MTD, four sets dedicated to LRU (0, 5, 10, 15), another four sets dedicated to BIP and the follower sets apply which policy between LRU and BIP are decided by PS. As already mentioned, if there are a miss incurred in the sets dedicated to LRU increases PS, and a miss incurred in the sets dedicated to BIP decreases PS. If MSB of PS is 1, then MTD uses BIP, otherwise MTD uses LRU.

B. Analysis of DRRIP

Re-Reference Interval Prediction (RRIP) represents the order in which blocks are predicted to be re-referenced [13]. The block of RRIP chain is predicted to have a near immediate re-reference interval which means a cache block will be re-referenced soon, while the block at the tail of the RRIP chain is predicted to have a distant re-reference interval implies that a cache block will be re-referenced in a long time. DRRIP can dynamically choose between SRRIP and BRRIP depending on which policy incurs fewer misses.

1. SRRIP

In order to prevent blocks with a distant re-reference interval from polluting the cache, SRRIP uses M bits in per cache block to store one of 2^M possible Re-Reference Prediction Values (RRPV) which can dynamically learns re-reference information. For all experiments, we set the value of M is 2. The value of RRPV is '0' implies that a block was recently used and the block is predicted to be re-referenced in the near immediate future. The value of RRPV is '3' implies that a block was not recently used and the block is predicted to be re-referenced in the distance future. Otherwise the value of RRPV is '1' or '2' implies a long re-reference interval.

Firstly, the value of RRPV in all the cache blocks is 3. On cache fills, SRRIP always predicted that the missing block will have a near immediate re-reference. On a cache hit, set RRPV of block to '0'. On a cache miss, SRRIP selects the victim cache block whose predicted re-reference is in the distance future by finding the first RRPV of block is 3, then replace block and set RRPV to '2'. If there is no RRPV of block equals to '3', increment all RRPVs then continue to do this operation until find the RRPV of block equal '3'. Figure 2 illustrates the behaviour of SRRIP using 4-entry cache initialized with invalid blocks 'I'.

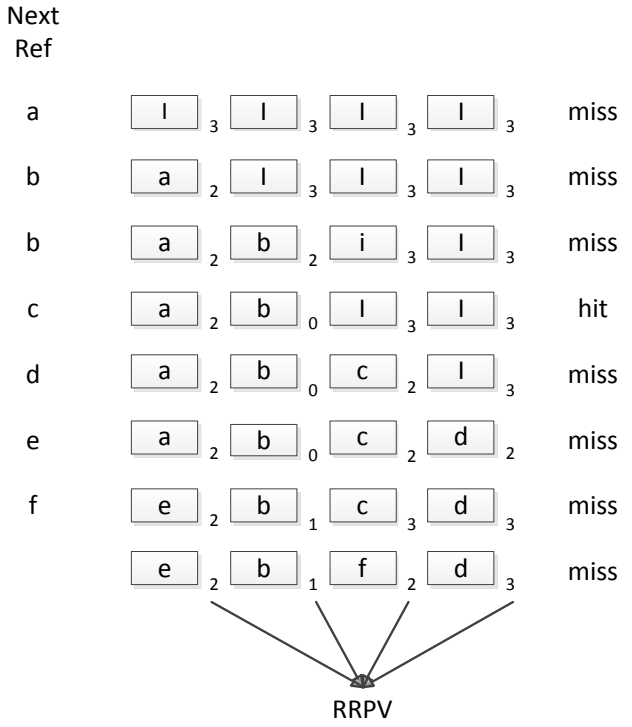

Figure 2: Behaviour of SPPIR for a Mixed Access Pattern

From the Figure 2 we can see the exact process of SRRIP clearly. SRRIP can automatically adapt to changes in the application working set size and thus can improve performance significantly.

2. DRRIP

BRRIP is promoted in Reference [13] in order to avoid cache thrashing and adapt different kinds of working sets. Similar to BIP, BRRIP has a parameter ε which controls the percentage of incoming lines that are replaced the block which RRPV is '2'. We set ε = 1/32, namely replacing incoming

lines to the block whose RRPV is '3' 31 times and only one time to the block whose RRPV is '2'.

For non-thrashing access patterns, always using BRRIP is unprofitable. Therefore we use Set Dueling to dynamically choose between SRRIP and BRRIP depending on which policy incurs fewer misses. The concrete implementation is described in section 2. We only need 2 bits to detect what kind of replacement policy each set needed without requiring significant hardware overhead. The schematic of DRRIP is shown in Figure 3.

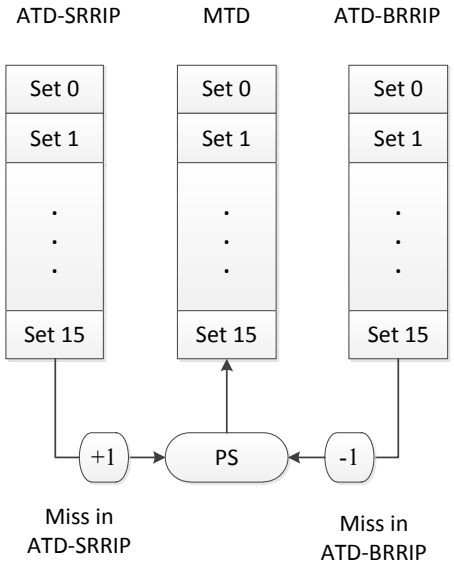

Figure 3: The schema of DRRIP [1]

For all experiments in this article the value of N is 16, namely a cache containing 16 sets, and K is 4. PS can keep track of which ATD-SRRIP and ATD-BRRIP incurs fewer misses. If there are a miss in ATD-SRRIP, PS increases 1 and a miss in ATD-BRRIP decreases PS. MSB of PS is an indicator of which of the two policies incurs fewer misses. If MSB of PS is 1, then MTD uses BRRIP, otherwise MTD uses SRRIP.

III. Experimental Methodology

A. Simulator

In order to compare different cache replacement algorithms, we use the CRC which can provide the competitors with a trace driven simulation environment [12]. CRC provides access to the class structure holding meta-data and functions responsible for selecting the evicted lines in each LLC set. And the competition can be subdivided into 2 phases: a single and multi-core execution. In each phase, submissions are judged by the performance of the LLC replacement algorithm. Each submission is limited to 8 bits per cache line as well as 1 Kb of global memory.

B. Configuration of CRC

The RADEME in CRC has a detailed description of the specific configuration of the storage structure. The LLC configuration maintains 16 way sets and 64 bytes lines regardless of the number of cores, and follows for 1 MB per core. A signal core configuration with a 1MB LLC, and a 4-core configuration with a 4MB shared LLC. The size of L1 cache is 32KB, and L2 cache is 256KB. The specific configuration of LLC is shown in Table 1.

Table 1
Cache Configuration

Cache Size	Line Size	Associativity	Sets	Threads
1024KB	64B	16	1024	1

In the experiments, signal core has three level caches. We mainly study the cache miss rate of LLC. We use the SPLASH-2 as our benchmarks.

IV. Test Result and Experimental Analysis

A. Analysis of the Experimental Results

It is very necessary to compare the cache miss rate in order to assess the performance of the replacement policies. The miss rate illustrates the performance of the storage system and calculates the average memory access time. This article is based on signal core architecture to compare the pros and cons of each replacement policy. LRU and RANDOM policy is offered by the simulator, we achieved DIP and DRRIP based on CRC. The experiment results are shown in Table 2.

Table 2
Comparison of Different Replacement Policies Cache Miss Rate

Benchmark\ Cache miss rate	LRU	RANDOM	DIP	DRRIP
Ls	99.673	99.673	99.673	99.673
FFT	98.6264	98.6264	98.6264	98.6264
RADIX	83.2914	71.888	69.256	70.853
BARNES	72.8731	68.2726	66.736	66.9067

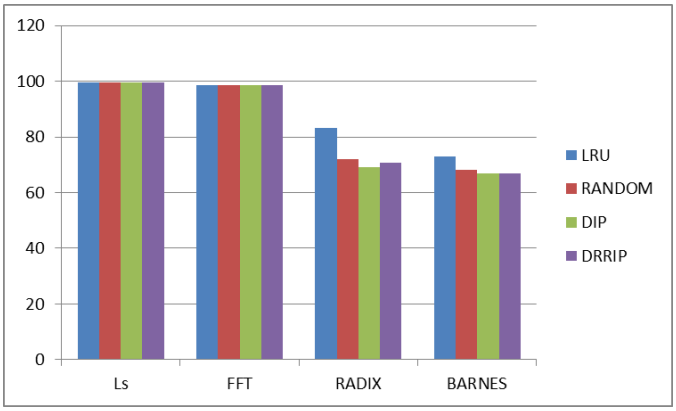

Figure 4: Comparison of Different Replacement Policies Cache Miss Rate

Figure 4 shows that less cache miss rate are obtained by DIP and DRRIP than traditional policies LRU and RANDOM.

In the best situation, DIP can decrease the cache miss rate by 16.85% and DRRIP is 14.94%. Because of DIP and DRRIP overcome the inefficient use of cache space as most of the lines installed occupy cache space without contributing to cache hits. LRU has bad performance because two reasons. Firstly, the line has no temporal locality which means the line is never re-referenced. It is not beneficial to insert such lines in the cache. Secondly, the line is re-referenced at a distance greater than the cache size, which causes the LRU policy to evict the line before it gets reused.

From the above experiments, we can see DIP can decrease the cache miss rate by 12.64% on average, and DRRIP can decrease the cache miss rate by 11.57% on average. This illustrates our LLC replacement policies is effective to adapt thrashing workloads and normal workloads, also we can see that different kinds of replacement policies have different performance for a particular benchmark because the desired optimization has intimately relationship with program characteristics. In order to solve the problem caused by LRU, we achieved the DIP and DRRIP to dynamically choose the best policy to apply in next cache line by taking the prediction ideology. This ideology is based on the Set Dueling mechanism which dedicates a few sets of the cache to each of the two competing policies and uses the policy that performs better on the dedicated sets for the remaining follower sets.

B. Hardware Overhead and Design Changes

BIP only requires a 5 bits counter to control the percentage of incoming lines that are placed in the MRU position due to we set $\varepsilon = 1/32$. DIP requires the per-thread 10 bits PS counter for choosing between LRU and DIP. Set Dueling does not require an additional storage, except for a single saturating counter. By way of comparison, SRRIP requires 2 bits register per cache block, and 4 logics to find pointers to the first '0', '1', '2', and '3' RRPV registers. In the event that a RRIP of block equals '3' is not found, RRIP also requires additional logic to add all the RRPV registers in the set. BRRIP is similar to SRRIP, but requires a 5 bits counter to control the percentage of incoming lines that are placed in the position which RRPV equals '2'. On this basis, DRRIP requires the per-thread 10 bits PS counter for choosing better policy.

From above compassion, we can see DRRIP requires more overhead than DIP, but the design changes for DRRIP is not on the critical path and thus do not affect the cache access time.

V. CONCLUSIONS

In this work, we advance the state-of-the-art LLC replacement policies via implementing two kinds of policies in CRC. The DIP and DRRIP are valuable to solve the problem that LRU always discards the least recently used item, and dynamically choose a better policy depending on which policy incurs fewer misses. With simulation testing on CRC, we found that the cache miss rate is decreased significantly compared with traditional LRU and RANDOM policy for the same benchmark FFT, RADIX and BARNES in SPLASH-2. Extensive experiments show cache miss rate can be mostly decreased from 88.29% to 69.26% in the signal core case. We also analysed the hardware overhead and design changes between DIP and DRRIP, and the characteristics of benchmark. The results verify that different kinds of replacement policies have different performance for a particular benchmark, and we can select the best average performance of the policy to get a desired optimization.

ACKNOWLEDGEMENT

Here and now, we would like to extend our sincere thanks to all those who have helped us make this article possible and better, and thank all lovely friends, for their encouragement and support. This work is supported in part by the Specialised Research Fund for the Doctoral Program of Higher Education of China (Grant No.20116102120049), the Natural Science Foundation of Shaanxi Province, China (Grant No.2013JQ8034) and the Basic Research Foundation of NWPU (Grant No.JC20120239).

REFERENCES

[1] R. Subramanian, "Adaptive Caches: Effective Shaping of Cache Behavior to Workloads", MICRO-28, 1995.

[2] Zhao Jianmin, and Zhao Jianhua, "An Optimal Replacement Policy for a System with Finite Spares", Journal of Systems Engineering and Electronics, 1999, in Chinese.

[3] Steven P. Vanderwiel, and David J. Lilja, "Data Prefetch Mechanisms", ACM Computing Surveys (CSUR), 2000.

[4] W. A. Wong, and J. -L. Baer, "Modified LRU Policies for Improving Second-level Cache Behavior", ACHPCA-6, 2000.

[5] Zhuang Weiqiang, Hu min, and Wang Dingxing, "Replacement Policy for Caching World-Wide Web Documents Based on Site-Graph Model", Tsinghua Science and Electronics, 2001, in Chinese.

[6] W. Lin, C. Fide, and B. Falsafi, "Predicting Last-touch References under Optimal Replacement", Technical Report of Michigan University, 2002，CSE-TR-447-02.

[7] N. Megiddo, and D. S. Modha, "ARC: A Self-tuning, Low Overhead Replacement Cache", Proceeding of the 2nd USENIX Conference on File and Storage Technologies, 2003.

[8] M. K. Qureshi, D. Thompson, and Y. N. Patt, "The V-Way Cache: Demand Base Associativity via Global Replacement", ISCA, 2005, (32):544-555.

[9] Nai Zhengbian, and Hao Chen, "A Least Grade Page Replacement Algorithm for Web Cache Optimization", Proceedings of the First International Workshop on Knowledge Discovery and Data Mining, 2008.

[10] Wang Deli, and Gao Deyuan, "A Sharing-Aware Active Pushing Cache Technology on Chip-Multiprocessor [J]", Journal of Xi'an Jiao Tong University, 2010, 44(10):18-23, in Chinese.

[11] Kahkashan Tabassum, Asia Sultana, and A. Damodaram, "A Heuristic Cache Replacement Policy for Data Caching", Proceeding of 2010 4th International Conference on Intelligent Information Technology Application, 2010.

[12] Yuval Peress, Ian Finlayson, and Dr. Gary Tyson, "CRC: Protected LRU Algorithm", JILP Worshop on Computer Architecture Competitions, 2010.

[13] A. Jaleel, K. B. Theobald, and S. C. Steely Jr., "High Performance Cache Replacement Using Re-reference Interval Prediction (RRIP)", Proceedings of the 38th International Symposium on Computer Architecture, 2010

[14] Carole_Jean Wu, Aamer Jaleel Margaret Martonosi, and Simon C. Steely, "PACMan: Prefetch-Aware Cache Management for High Performance Caching", Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, 2011.

[15] Li Chongming, Wang Dongshen, and Xue Yibo, "Enhanced Adaptive Insertion Policy for Shared Caches", Proceedings of the 9th international Conference on Advanced Parallel Processing Technologies, 2011.

[16] Tan Lin, Yang Jian, and Cheng Zhijun, "Optimal Replacement Policy for Cold Standby System", Chinese Journal of Mechanical Engineering, 2011, in Chinese.

[17] Shan Ding, and Li Yuanyuan, "LRU2_MRU Collaborative Cache Replacement Algorithm on Multi-core System [A]", Proceedings of 2012 IEEE International Conference on Computer Science and Automation Engineering, 2012，23(1):14-17, in Chinese.