Cache Replacement Using Re-reference Interval Prediction with Adaptive Bypassing (RRIPAB)

Man-Man PENG¹,a,*, Peng WANG¹

¹College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, China
a pengmanman778@sina.com
*Corresponding author

Keywords: Replacement, Adaptive Bypassing, Useless Blocks.

Abstract. In this paper we present a new policy to improve cache performance. The new algorithm called Cache Replacement Using Re-Reference Interval Prediction With Adaptive Bypassing (RRIPAB) which combines the Re-Reference Interval Prediction (RRIP) with a adaptive bypassing algorithm. RRIP have two advantages. The first one is can move block quickly which can reduce the time of useless blocks stay in the cache. The other one is the new block remains in the cache in a shorter time which can evict new blocks fast and give them the time to train. There also cause a problem that many block will be evicted before it hit. So, adaptive bypassing algorithm is a solution of this problem. We evaluate the RRIPAB by performing a simulation study. The simulation results show that RRIPAB reduces the cache miss count by comparing to the LRU policy, the Dynamic Re-Reference Interval Prediction (DRRIP) and the Dueling Segmented LRU Replacement Algorithm with Adaptive Bypassing about 8.7%, 3.4% and 3.5% in 1M LLC, and about 9.9%, 4.4% and 2.2% in 4M LLC.

Introduction

An optimal replacement policy can retain block which will be reused in the cache, and evict the useless block quickly. Practical cache replacement policies mainly consist of three parts, promotion, eviction and insertion.

Commonly we use Least Recently Used (LRU) policy, the LRU chain represents the recency of cache blocks referenced. The MRU position represents a block that was most recently used while the LRU position represents a block that was least recently used. When a block hit, we need change its position in the LRU chain, the process means promotion. When cache misses, we need select a block to evict and insert a new block in a new position, the process means eviction and insertion respectively.

There are three classical algorithms, LRU, FIFO and NRU. LRU is the Least Recently Used policy. When cache hits, LRU promotes the block to the MRU position. When cache misses, LRU evicts the block in the LRU position and inserts a new block into the MRU position. FIFO is first in first out policy. Whatever cache hits or not, the policy do nothing about promotion. When cache misses, FIFO evicts the first block get into the cache and inserts a new block which called last in. NRU is not the Least Recently Used policy. The policy use a bit named nru to divide the cache into two sections, near-immediate re-reference interval and distant re-reference interval. If nru is set, it means the block is not used in recently, and if it is not set, the block will be used in recently. When cache hits, the nru of the block will not be set. When cache misses, we need choose a block in the distant re-reference interval, and need insert a new block into the near-immediate re-reference.

Many algorithms develop from the three policies, in which the most is LRU. The Dynamic Insertion Policy (DIP) is a policy of insertion. The promotion and eviction of the DIP are the same as LRU. The DIP is consist of two policies, the LRU and the BIP. The BIP inserts new cache block into the LRU position or MRU position, which is decided by a random mechanism. The DIP uses
set-dueling to decide the follower sets use LRU or BIP [2]. Insertion Policy Selection Using Decision Tree Analysis, it is a classical policy which combines LRU and insertion. we need five leader sets correspond to five inserted position, LRU, near LRU, Middle, near MRU, MRU respectively. The five leader sets use set-dueling to decide the follower sets inserted position [3]. Some policy develop from FIFO, like Pseudo-LIFO [4]. the most famous cache replacement policy come from NRU is RRIP. The RRIP use M bits to sign each block, which divides the cache into three sections, near-immediate re-reference interval, intermediate re-reference interval and distant re-reference interval. The policy inserts new blocks into intermediate re-reference interval, evicts blocks from the distant re-reference interval, and promotes block into near-immediate re-reference interval. The RRIP have two advantages, first one is it can move blocks more fast in the cache, the other one is it can train new inserted block better. Due to the shorter train time, more blocks which will be used in the near future are evicted. We can use some methods to solve this problem. This method should be aimed at eviction, and should be provided with predicting usability of the block. So, the bypass may be a suitable method.

In this paper, we propose re-reference interval prediction with adaptive bypassing (RRIPAB). The RRIPAB is combined RRIP and adaptive bypassing that can eliminate eviction of some usable new blocks. Adaptive bypassing use bypass tracking logic implemented for some leader sets to decide whether or not bypass the cache. We use two policies implemented in two leader sets. The winner of set-dueling is implemented by follower sets.

The rest of this paper is organized as follows, Section 2 introduces the RRIPAB. Section 3 provides the experimental evaluation. Section 4 result and analysis.

Re-reference Interval Prediction with Adaptive Bypassing (RRIPAB)

This paper combines RRIP and adaptive bypassing. The RRIP can not only solve some problems that the useless cache block remains in the cache for a long time, also their own problem is lead some useful inserted block evicted Prematurely. The adaptive bypassing can predict which block will be used and which block will be useless. If the block predict to be useless, we can choose bypass the cache. So, we can use adaptive bypassing to solve the problem of RRIP.

Re-reference Interval Prediction

The RRIP is developed from the NRU. NRU just have one bit which called nru-bit to sign each block. The NRU divide the cache blocks into two sections, near-immediate re-reference and distant re-reference by nru-bit. An nru-bit value of '0' means that a block is recently used and the block is predicted to be re-reference in the near-immediate future. An nru-bit value of '1' means that a block is not recently used and the block predicted to be re-reference in the distant future. Because the NRU lack of other status, the new inserted blocks are evicted easier and the protection of blocks in the near-immediate re-reference is more vulnerable. So, Enhanced the granularity of the re-reference interval is necessary.

In order to increase the status of blocks, we can use M bit to sign every cache block. M bit creates status that means each cache block may have different status. Re-reference prediction value(RRPV) stores one of possible status. The cache blocks divide into three sections. The first one is the RRPV of zero which means the cache block is predicted to be re-reference in the near-immediate future. The second one is the RRPV of which means the cache block is predicted to be re-reference in the distant future. The other one is other status of RRPV called intermediate re-reference which means greater than near-immediate future but less than distant re-reference [5]. The more status can change cache replacement of NRU drastically. Firstly, we can insert new cache block into the intermediate re-reference interval that makes new blocks have more time to train themselves. Secondly, blocks in the near-immediate re-reference interval move to the distant re-reference interval will take more time.
Adaptive Bypassing

Adaptive bypassing is an algorithm that can decide whether or not to evict the block in the LRU position when cache misses. If decide not to evict the victim block, it will bypass the cache. The decision on bypass the cache is made by bypass tracking logic implemented for each set in the cache. The logic evaluates the effect on cache performance of bypass decision. When cache misses, we can choose to bypass the cache and keep the victim cache block or replace the selected victim cache block and insert a new block. When a block is selected to bypass the cache, we will assess the impact of the decision of bypass by storing the tag of the block which refer to as competitor tag. In addition, we also store the pointer to the tag of the victim block which refer to as competitor way. A reference to the competitor way indicates effective bypass. Conversely, a reference to the competitor tag indicates ineffective bypass. Effective or ineffective bypass decide increase or decrease the probability of bypass. Each effective bypass can doubles the probability which will decide future bypass. And, ineffective can halves the probability of future bypass [6].

Like other algorithms, we also use set-dueling to choose the better method in the run time. The policy selector uses two group sampled sets, one group sets use RRIP policy and the other group sets use RRIP coupled with adaptive bypassing. We add an auxiliary tag directory (ATD) to each of these sets and concurrently implement both dueling policies (policy A, policy B) for these sets. We also use a saturating counter to track the performance of the two policies in the sampled sets. The counter updates just when the two policies have different results. If policy A hits and policy B misses, the counter is decremented. Conversely, if policy A misses and policy B hits, the counter is incremented. We propose cache replacement algorithm is RRIP coupled with adaptive bypassing. The RRIPAB can eliminate eviction of some usable new inserted cache block. Adaptive bypassing can reduce overhead of some new inserted block which just use only once.

Evaluation

We evaluate the RRIPAB within the CRC framework. We use SPEC CPU2006 as the benchmark for our evaluation. Each benchmark is compiled by using GCC.3.4.6 with the"-O3 -fomit-frame-pointer -funroll-all-loops" option. To collect traces for a snippet of an application, we use the -fwd and -icount options to the pin tool. The simulation warms them up for the first 40 billion instructions and generate a trace for 100 million instructions. We use ref inputs for the simulation. In order to get good results, we just use 15 workloads which is sensitive to this policies. As competitive replacement policies, we also evaluate segmented LRU replacement with adaptive bypassing (LRUAB) and DRRIP in a single-core environment.

For a 16-way associative cache, the base RRIP policy need M bits per cache block. In this paper, the M equals 3. So, we need 3 bits per cache block. For adaptive bypassing logic requires 22 bits per set. Firstly, the tag of competitor needs 16 bits which use to store lower 16 tag bits of the track block. Secondly, 16-way associative cache need 4 bits to sign each way. So, we need 4 bits point to the competitor way. Thirdly, we need two 1-bit to record the tracking status. The cost of selector policy is mainly consumed by ADT. The function of ADT is tracking sampled sets which is one of every 32 sets. Each sampled set has separated tags and replacement stales for each policy. Separated tags need 16 tag bits and 1 valid bit per cache block. Replacement stales need 102 bits per set for each policy. We also need some registers to adaptive bypassing. Saturating counter needs 11 bits, and as the random number generator, Linear Feedback Shift Register needs 32 bits. Each policy needs probability register which need 4 bits. All of that, we need 93.4k.

Results and Analysis

In order to validate our cache replacement algorithm, we choose 17 SPEC CPU 2006 traces with different inputs and simulation points which are sensitive to different cache replacement algorithms. we implement our experiment increase LLC from 1M to 4M, and compare our policy with LRU, RRIP and LRU with adaptive bypassing.
We mainly measured LLC miss rate (MPKI) to evaluate the policy. Figure 1 shows that MPKI reductions of the other three policies compare to LRU. As we can see from the Figure 1, RRIP, LRU with adaptive bypassing (LRUAB) and RRIP with bypassing (RRIPAB) compare to LRU, the MPKI reductions about 5.3%, 5.2% and 8.7%. It is means the MPKI of RRIP with adaptive bypassing reduces about 3.4% and 3.5% compare to the other two policies. Figure 2 shows MPKI reductions of 4M LLC. Compare to LRU the other three policies MPKI reductions are 5.5%, 7.7% and 9.9%. It is also means the MPKI of RRIP with adaptive bypassing reduces about 4.4% and 2.2% compare to the other two policies.

![Figure 1. The MPKI of 1M LLC.](image1)

![Figure 2. The MPKI of 4M LLC.](image2)

References