COMPUTER ARCHITECTURE - 2

ASSIGNMENT -2

NAME: Vasundhara Venkata Krishna

ASU ID: 1218418357

**Introduction to REFCOUNT cache replacement policy**

The ‘Least recently used’ cache replacement policy is based on when the data was last accessed or touched. If there is a hit, then the accessed block is assigned as MRU(Most recently used). If there is a miss then the LRU block is replaced with the block that caused a miss and makes it MRU. But in recent studies, it has been shown that in a highly associative cache, the gap between the theoretical optimal replacement policies and LRU is very large. The counter-based(reference counter) cache replacement policy is an approach which significantly improves highly associative cache replacement performance.

In the Refcount cache replacement policy, each of the blocks in a set is given an Event Counter. The Event Counters of all the blocks are incremented whenever there is an event occurring at that set. And when the counter exceeds a given threshold, then that block is selected to be replaced. The refcount cache replacement policy predicts the blocks that will not be reused and evicts them.

There are two different types of reference counters based on the type of intervals during which the event counter is incremented. The AIP(Access Interval Prediction) records the number of access to the set where the line resides, between consecutive access to the line. The LvP records the number of accesses to itself during the interval in which the line resides continuously in the cache. The event count threshold is selected conservatively as the maximum of such numbers. In both these algorithms, the cache lines are given an extra 21 bits to store prediction information and an additional 40Kb tagless direct-mapped prediction table is added between L2 cache and its lower memory. In this assignment, we are implementing the AIP algorithm.

In the AIP algorithm, the counter of each block is incremented each time there is access to the set. If the access is a hit, then the maximum counter value is assigned to MaxCpresent. If MaxCpresent is larger than any of the counter values, then no change is made. The counter value of the block which was a hit is made zero. If the access is a miss, then we have to find a block that can be evicted. This is done by checking if the counter value of that block is greater than the MaxCpresnt Value and MaxCpast value and if the Confidence bit is 1. The confidence bit is made 1 whenever MaxCpresent value becomes equal to MaxCpast value. There might be a case where there is more than one block which has expired. In that case, the block with the highest counter value is evicted. There might also be a case where there is no expired block. In this situation, the LRU block is chosen to be evicted. Once the block to be evicted or the victim is chosen, the prediction table is updated with its MaxCpresent and conf bit values. And then the new block is placed into the cache after making its counter and MaxCpresent value to zero and assigning the MaxCstored value as its MaxCpast value and ConfStored value as its current confidence bit value.

The Refcounter cache replacement policy sped up 10out of 21 Spec2000 applications by up to 40% and the average improvement is 11%. The refcount cache replacement policy outperforms other predictors both in accuracy and coverage(fraction of total evictions that are initiated by the predictors) while using simpler hardware support.

1. Fluidanimate 1MB and 4MB:

|  |  |  |
| --- | --- | --- |
| L2 Cache Size | LRU Replacement Policy | REFCOUNT Replacement Policy |
| 1MB | 56.9813 | 60.0607 |
| 4MB | 48.0429 | 51.9593 |

1. Bodytrack 1MB and 4MB:

|  |  |  |
| --- | --- | --- |
| L2 Cache Size | LRU Replacement Policy | REFCOUNT Replacement Policy |
| 1MB | 13.3891 | 16.3913 |
| 4MB | 9.9800 | 11.1369 |