LRU and Pseudo-LRU

Pseudo-LRU is an approximate LRU algorithm which requires much less overhead to manage. Thus pseudo-LRU may be less effective and cause more misses than LRU but the update of the bits required to identify the victim block is much less complex.

LRU is easily manageable for a 2-way cache. One single bit per set is sufficient to point to the LRU line. However, for larger set sizes, the complexity of LRU grows exponentially. For a 4-way cache, exact LRU needs four two bits fields, which must be updated on each access. Each two bit field is associated with a line and indicates the relative recency of access to each line in the set. This is a total of 8 bits. For a 16-way cache, exact LRU will require 16 sets of 4 bits each or 64bits total. In general, for an N-way cache, exact LRU needs $N\log_2 N$ bits.

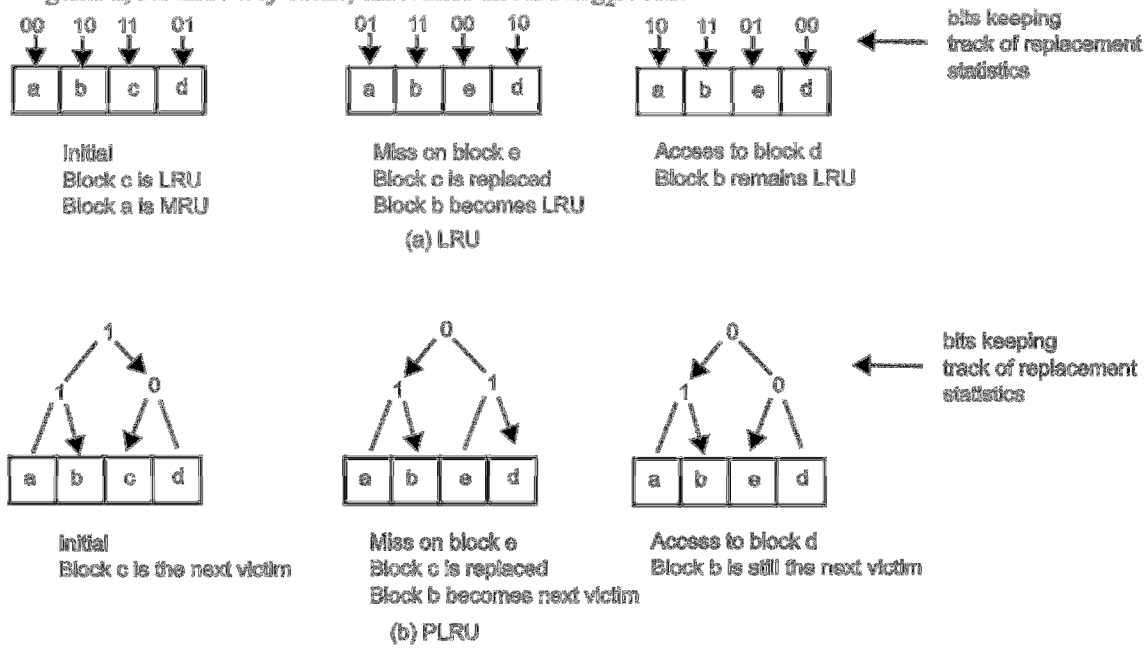

Figure 3. LRU vs. PLRU

For this reason pseudo-LRU is often preferred. Pseudo-LRU only works with power-of-two set sizes. In pseudo-LRU, the lines in each set are recursively partitioned in two subsets, and one bit points to the LRU subsets at each level of the recursive partition.

Let's say that we have N lines in a set, where $N=2^n$. These N lines are split into two subsets of size N/2 each. One bit is enough to track the LRU subset. Then each subset is again split into two sub-subsets of size N/4 each, which requires two bits to track the LRU sub-subsets in the subsets, for a total of 3 bits. Next we split the sub-subsets again into sub-sub-subsets of size N/8 each, which requires four bits, for a total of 7 bits. We do this recursively until we cannot split anymore because we have reached individual lines. The total number of bits is $\log_2 N-1$, much less than for LRU. For example, for a 16-way cache, the number of bits is 15, as compared to 64 for LRU. Therefore the FSM updating the replacement policy bits is much simpler.

When the set size is 2, LRU and pseudo-LRU are identical. Only 1 bit is needed. When it is more than 2 lines, the bit at each level points to the LRU subset at the next level. So we keep track of the LRU subset at each level.

The difference between LRU bits and pseudo-LRU (PLRU) bits management is illustrated in Figure 3 for a set size of 4 cache lines (to simplify, the cache size is four lines).

The four lines are divided into two subsets: S0={line0,line1} and S1={line2,line3}. Each subset is subdivided into two lines. In the initial state, S0 contains blocks a and b and S1 contains blocks c and d. At the root of the tree bit value 0 points to S0 and bit value 1 points to S1. This correspondence is indicated by the arrows. Within S0, one bit points to line0 (value 0) or line1 (value 1). A bit always points away from the MRU subset and towards the LRU subset. So in the initial state, the MRU subset is S0 and the LRU subset is S1. Within S1, the LRU line is line2 (containing block c) and the MRU line is line 3. This is consistent with the initial state of LRU. After a miss on block e the victim is pointed to by the arrows, i.e., block c contained in line2. At this point the LRU subset becomes S0 and line1 is the new candidate for replacement. Then block d is accessed. S1 remains the MRU subset and line1 contains the next victim, i.e., block b. This is still consistent with LRU.

PLRU approximates LRU, and in this example it points to the same victim as LRU. However, PLRU will eventually diverge from LRU.