Cache Replacement Policy

王琛

Classical Replacement Policy

- LRU
- MRU
- LIP: LRU position -> MRU position
- BIP: random a position between LRU and MRU
- Protected LRU: protect blocks with high reference

Recently Proposed Policy - RRC

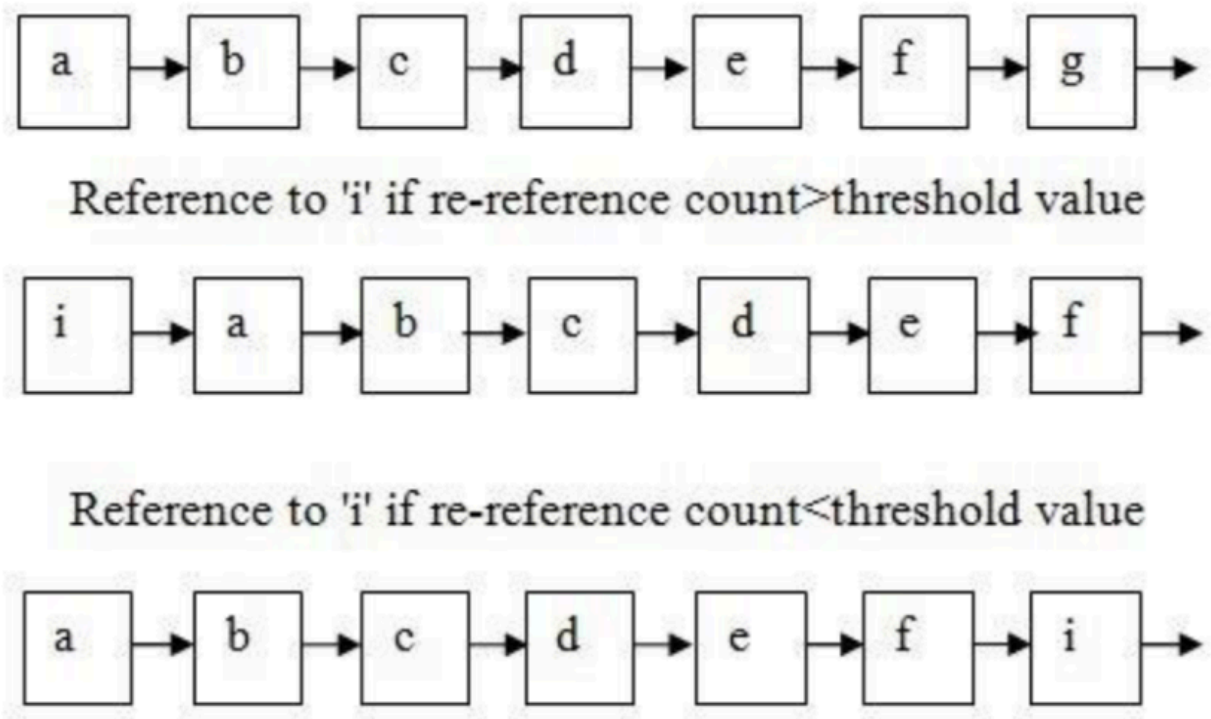
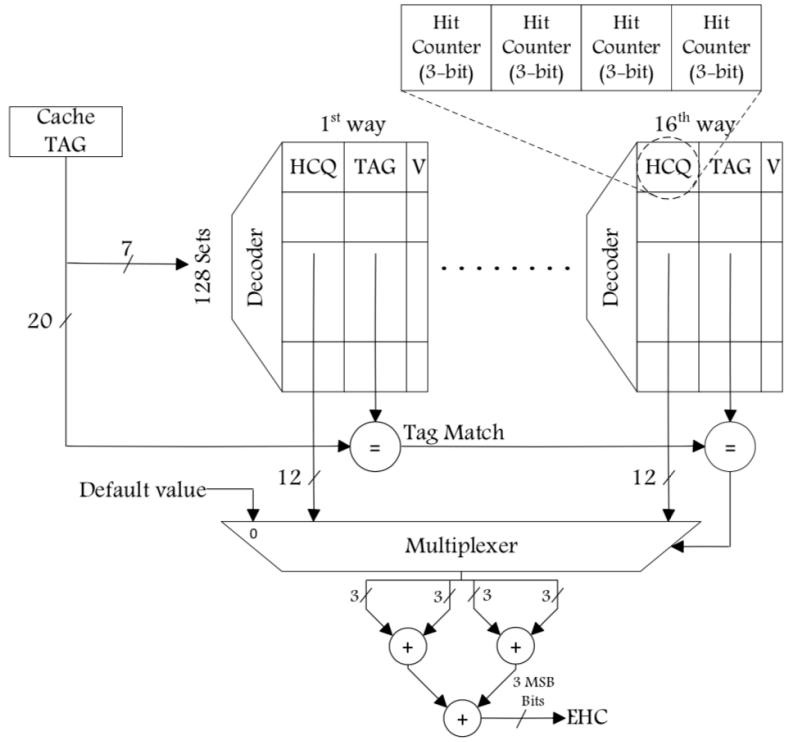


Fig. 2. New policy based on re-reference count

Recently Proposed Policy - EHC

- Re-reference count is related to re-reference distance
- Choose victim with maximum re-reference distance



Score

- Assign a score for each block
- Initial score is half of maximum score
- Randomly choose from victims whose scores are below threshold
- When a block is hit, increase its score and decrease others
- Score is just another way of representing the position in queue

Summary of Current Policies

- The initial position
- Change position based on recent behavior

New Policy - WMBP

- WMBP – WorkSet Miss-Rate based points
  - Choose victim and initial points based on miss rate
  - High miss-rate indicates higher probability of changing working area

- Improvement of Score
  - Change of score and the threshold is an absolute value

WMBP – Initial Points

- Higher miss-rate -> Closer to MRU -> Higher point

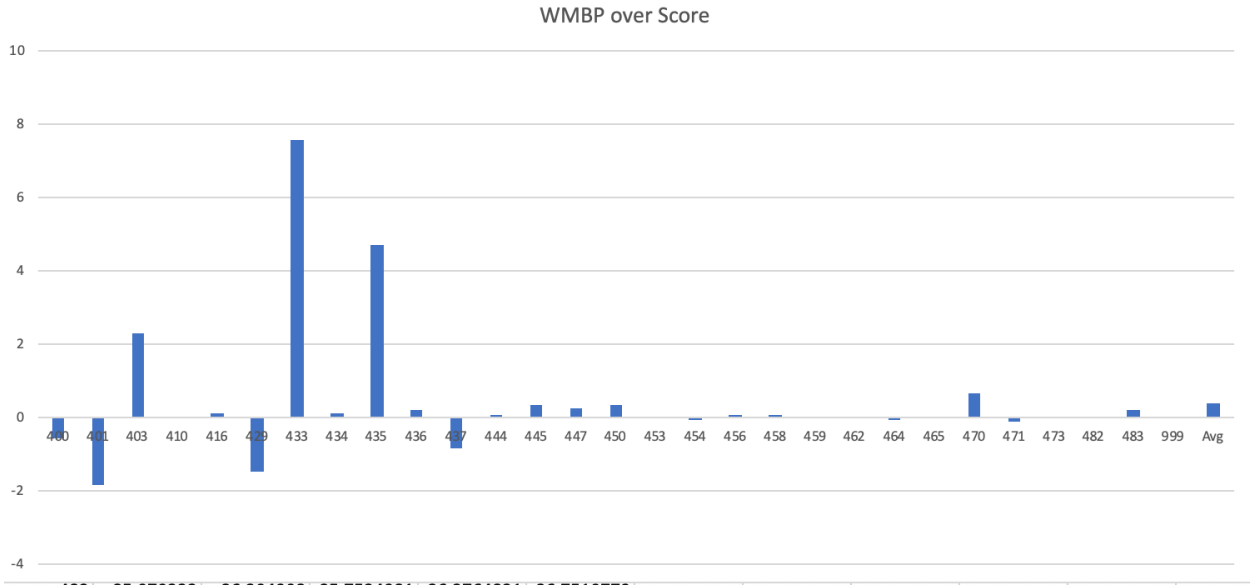
```
if(cacheHit == false) {
    if(5*numAccess > WINDOW_SIZE) {
        float missRate = float(curMiss)/float(numAccess);
        replSet[updateWayID].point = minPoint + int(missRate*(maxPoint-minPoint));
    } else {
        replSet[updateWayID].point = MAX_POINT >> 1;
    }
    ++curMiss;
} else {
    replSet[updateWayID].increase();
}
```

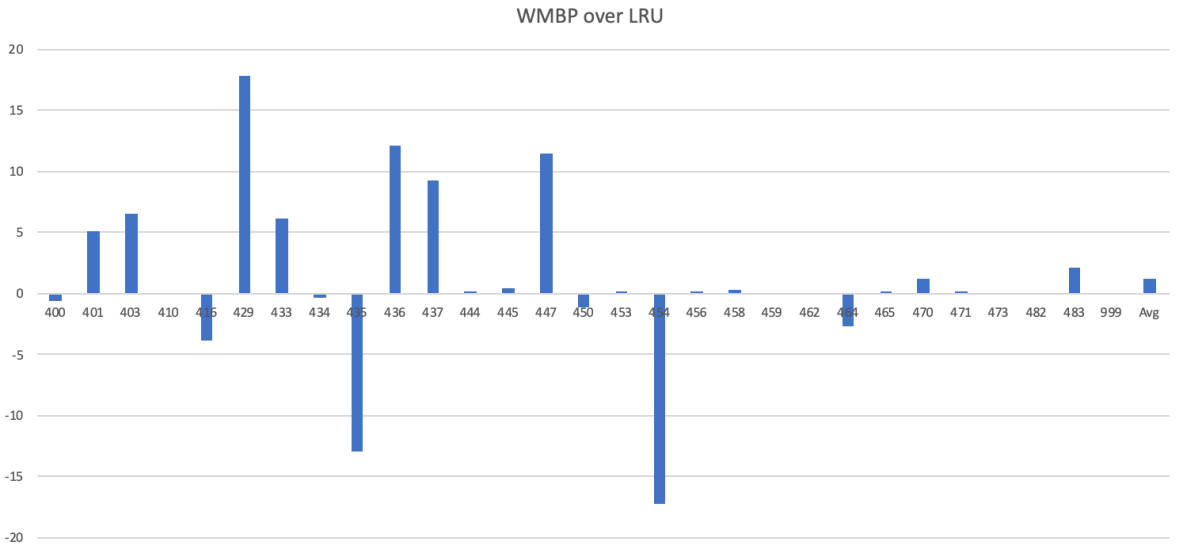
WMBP – Victim Choosing

- In Score, victims set is filtered with an absolute threshold.
- Higher Miss-rate -> More victims

```
if(5*numAccess > WINDOW_SIZE) {
    float missRate = float(curMiss)/float(numAccess);
    if(missRate > 0.5) {
        victimNum += missRate * 4;
    } else {
        victimNum -= missRate * 4;
    }
    if(victimNum > VICTIM_SET_MAXSIZE) victimNum = VICTIM_SET_MAXSIZE;
    if(victimNum < 1) victimNum = 1;
}
```

WMBP - Result


WMBP over Score
10
8
6
4
2
0
-2
-4
400 401 403 410 416 429 433 434 435 436 437 444 445 447 450 453 454 456 458 459 462 464 465 470 471 473 482 483 999 Avg


WMBP over LRU
20
15
10
5
0
-5
-10
-15
-20
400 401 403 410 416 429 433 434 435 436 437 444 445 447 450 453 454 456 458 459 462 464 465 470 471 473 482 483 999 Avg

WMBP - Result

- Easily adapt to circular sequences like abcdeabcde….

```
int main(){
    int a[278528], i, j;
    for (i = 0; i < 10; i++) {
        for (j = 0; j < 278528; j++) {
            a[j] = i + j;
        }
    }
    return 0;
```

Policy	Miss Rate	Running Time
WMBP	30.74	25.49
Random	34.05	24.31
PLRU	51.99	26.71
LRU	99.83	28.85
Score	32.66	31.03

Discussion

- Strength
  - Dynamically change initial position
  - Solve the problem of LRU
- Weakness
  - Slow
  - Hard to implement on chips