

# **ECE7995 Caching and Prefetching Techniques in Computer Systems**

## Lecture 8: Buffer Cache in Main Memory (III)

# Facts about LRU-K

- Advantages:

- Scan-resistant: it gives low priority to pages that are just scanned or infrequently accessed, so as to keep really “hot” pages in the buffer.
- Flexible: it works well in a multitasking system without a priori knowledge about of programs’ access patterns.

- Disadvantages:

- Its time overhead is  $\log(N)$ , where  $N$  is number of pages in the buffer. (Each page access requires  $\log(N)$  work to manipulate a priority queue)
- It has tuning parameters:
  - ✓ Correlated Reference Period : when temporally close accessed occur to a page, they count as a single access
  - ✓ Retained Information Period : how much history to keep?

# What's the 2Q Replacement?

- Objective: 2Q attempts to create an algorithm that is as good as LRU-2, but has constant-time complexity and is self-tuning
  - 2Q attempts to improve LRU-2 by reducing its CPU overhead from logarithmic to constant
- Observations:
  - LRU works by removing cold pages from the buffer to make space for missed pages.
  - But if the missed is cold, a warmer page may be replace to make space for the cold one, and the cold page will reside in the buffer for a considerable amount of time.
- The idea for improvement of LRU is to test if a missed page is **sufficiently** hot before deciding to admit it for a long-term stay in the buffer.

Refer to this paper: *Theodore Johnson and Dennis Shasha, "2Q: a low overhead high performance buffer management replacement algorithm", 20th International Conference on Very Large Databases, 1994.*

## The Simplified 2Q Replacement Algorithm

Two queues: A1 and Am

- A1 contains recently accessed pages and is managed in FIFO fashion. This queue is used to test if a missed is sufficiently hot so as to deserve a long-term stay in the buffer.
- A page that is accessed again while it was in A1 (pass the test for its “hotness”) is promoted from A1 to Am. Am is managed with LRU.
- A page that is NOT accessed while it was in A1 is evicted from A1 (also from the buffer).

---

```
if p is on the Am queue
then
    put p on the front of the Am queue
    /* Am is managed as an LRU queue*/
else if p is on the A1 queue
then
    remove p from the A1 queue
    put p on the front of the Am queue
else /* first access we know about concerning p */
    /* find a free page slot for p */
    if there are free page slots available
    then
        put p in a free page slot
    else if A1's size is above a (tunable) threshold
        delete from the tail of A1
        put p in the freed page slot
    else
        delete from the tail of Am
        put p in the freed page slot
    end if
    put p on the front of the A1 queue
end if
```

## Improvement of 2Q

---

On accessing a page X :

```
begin
  if X is in Am then
    move X to the head of Am
  else if (X is in A1out) then
    reclaimfor(X)
    add X to the head of Am
  else if (X is in A1in) // do nothing
  else // X is in no queue
    reclaimfor(X)
    add X to the head of A1in
  end if
end
```

```
// If there is space, we give it to X.
// If there is no space, we free a page slot to
// make room for page X.
reclaimfor(page X)
begin
  if there are free page slots then
    put X into a free page slot
  else if(|A1in| > Kin)
    page out the tail of A1in, call it Y
    add identifier of Y to the head of A1out
    if(|A1out| > Kout)
      remove identifier of Z from
      the tail of A1out
    end if
    put X into the reclaimed page slot
  else
    page out the tail of Am, call it Y
    // do not put it on A1out; it hasn't been
    // accessed for a while
    put X into the reclaimed page slot
  end if
end
```

Pages in A1 that are accessed again during a short period of time should remain in A1 and not be promoted. (similar to the (CRP) Correlated Reference Period for LRU-K)

- Method: split A1 into A1in and A1out (size of A1in is  $K_{in}$ , and size of A1out is  $K_{out}$ )
  - ✓ A1in retains pages that were access once recently
  - ✓ A1out keeps track of pages recently evicted from A1in
  - ✓ If a page in A1in is accessed again, do nothing
  - ✓ If a page in A1out is accessed again, read it from disk and promote it to Am

## Unsolved Issue of 2Q

- **Tuning Parameters in 2Q:**
  - The authors say that the empirical setting ( $K_{in} = 25\%$  of memory slots and  $K_{out} = \text{identifiers for } 50\%$  of the memory slots) is a good choice in practice
- **The size of  $A_{in}$  ( $K_{in}$ )**
  - $K_{in}$  is determined by the correlated reference period.
  - If  $K_{in}$  is too small, it cannot cache correlated blocks. However, a large  $K_{in}$  would take away memory originally for  $A_m$ .
  - If CRP is known in advance, set  $K_{in}$  as small as CRP.
  - This parameter is relatively earlier to tune and isn't performance-critical.
- **The size of  $A_{out}$  ( $K_{out}$ )**
  - $K_{out}$  determines the criterion for admitting a re-accessed block into  $A_m$ . (How should a block be considered as sufficiently hot?)
  - $K_{out}$  should be determined by access pattern:  $K_{out}$  should be smaller with stronger temporal locality, and vice versa.
  - But  $K_{out}$  is a fixed parameter in 2Q.