Lab4Answers

Mengyue Hang (hangm@purdue.edu), 04/28/2017

Implementation

I have a global frame table `frametab` which holds all frames metadata (like frameid, owner pid, type - PAGE or DIR or ...), and a global variable `gca_head` which holds the current frame head in the loop (initialized to be at the **middle** of the frametab, which adds some randomness).

```
struct frame_t {
    int  id;
    pid32  pid;
    int  type;
    int  actual_dirty_bit;
    ...
}

frame_t frametab[NFRAMES];
frame_t gca_head = NFRAMES/2;
```

`actual_dirty_bit` is initialized to be 0;
`gca_head` is initially pointed to the `NFRAMES/2` th frame.

The GCA policy is implemented following the textbook: when we need to evict a frame, we loop in the frametab. For each valid frame (frame type = PAGE), we compute the virtual page number, and get the page entry in the corresponding page table. We check `pt_acc` and `pt_dirty` bits, when they are:

- (0, 0): evict the frame; make `gca_head` points to the next frame.
- (1, 0): mark `pt_acc` bit to be zero and bypass;
- (1, 1): mark `pt_dirty` bit to be zero, and make `actual_dirty_bit` for the frame to be 1. Bypass.

We keep looping the frametab, until there is one victim. The maximum number of accesses is (2*NFRAMES) + 1.

For the **dirty bit**, when we mark it to be zero, we actually save the value in the frame's `actual_dirty_bit` entry. When we need to check whether this page is dirty or not in other situations,

like swapping out, we actually do a `||` operation: `pt_dirty || frametab->actual_dirty_bit`. As `actual_dirty_bit` is initialized to be 0, and only set to 1 when the page is actually dirty, we can maintain the real value of the dirty bit.

FIFO v.s. GCA

Since FIFO doesn't check the `pt_acc` and `pt_dirty` bits, it will evict some dirty pages; while GCA checks the dirty bit first, which gives priority to those undirty pages when evicting. Same to access bit.

(It seems we don't need to talk about the performance any more, but anyway...)