Notes (Enhanced)

Sunday, October 9, 2016 3:55 PM

A nice reference for Enhanced 2nd Chance

http://courses.cs.tamu.edu/bart/cpsc410/Supplements/Slides/page-rep3.pdf

Additional notes on the Enhanced 2nd Chance (or Clock) Algorithm

(from "Operating Systems, Internals and Design Principles", Stallings, 8th edition)

The clock algorithm can be made more powerful by increasing the number of bits that it employs. In all processors that support paging, a ***modify*** bit is associated with every page in main memory and hence with every frame on main memory. This bit is needed so that when a page has been modified, it is not replaced until it has been written back into secondary memory. We can exploit this bit in the clock algorithm in the following way. If we take the use and modify bits into account, each frame falls into one of four categories:

- Not accessed recently, not modified ($u = 0; m = 0$)
- Accessed recently, not modified ($u = 1; m = 0$)
- Not accessed recently, modified ($u = 0; m = 1$)
- Accessed recently, modified ($u = 1; m = 1$)

With this classification, the clock algorithm performs as follows:

1. Beginning at the current position of the pointer, scan the frame buffer. During this scan, make no changes to the use bit. The first frame encountered with ($u = 0; m = 0$) is selected for replacement.
2. If step 1 fails, scan again, looking for the frame with ($u = 0; m = 1$). The first such frame encountered is selected for replacement. During this scan, set the use bit to 0 on each frame that is bypassed.
3. If step 2 fails, the pointer should have returned to its original position and all of the frames in the set will have a use bit of 0. Repeat step 1 and, if necessary, step 2. This time, a frame will be found for the replacement.

In summary, the page replacement algorithm cycles through all of the pages in the buffer looking for one that has not been modified since being brought in and has not been accessed recently. Such a page is a good bet for replacement and has the advantage that, because it is unmodified, it does not need to be written back out to secondary memory. If no candidate page is found in the first sweep, the algorithm cycles through the buffer again, looking for a modified page that has not been accessed recently. Even though such a page must be written out to be replaced, because of the principle of locality, it may not be needed again anytime soon. If this second pass fails, all of the frames in the buffer are marked as having not been accessed recently and a third sweep is performed.