CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Page Buffering

There are some efficiency improvements OS can make to page replacement regardless of which page-replacement algorithm is used

Goal is to either decrease reads and writes to memory or move them to more convenient time

Free-frame pool

First enhancement is to keep pool of free frames

This is different from free frames list because we keep frames free *even when it requires paging*

We are essentially decreasing size of our memory slightly to ensure some frames are always free

How can that be helpful?

When page fault occurs that mandates page replacement, victim page still selected to be replaced

However, new page is read into memory *before* old page is written to disk

This is possible because we have pool of free frames that new page can be read into without delay

New page *not* read into frame of page marked for replacement

Process can resume as soon as new page is read into free frame

Victim page written to disk after process resumes

Frame corresponding to victim page added to free frame pool, thereby keeping size consistent

Overall, no less I/O performed, but process can be restarted more quickly

Modified-page list

Another idea for moving I/O to more convenient time is using modified-page list

OS keeps track of which pages have been modified and would therefore be more expensive to page out to disk

Whenever paging device is idle, modified page from list is written to disk

Essentially no extra cost because system is using resources that were otherwise free

If page needs to be replaced later, it will only take half the time because outgoing page will not need to be written to disk

If page written to again, modified bit must be set again and page cannot be removed without writing to disk

Once again, this method moves I/O to more convenient time but does not decrease I/O

Free frame list and removed pages

Final scheme has potential to reduce disk I/O

Free frame list is maintained as before

Information about page previously held in free frame also tracked

If page A leaves memory, its frame will be sent to free frame pool but not necessarily written to

If page A comes back into memory, it is possible previously-used frame has not been chosen for another page

If frame is still in free pool, data still resides in frame, is available immediately, and does not need to be read back in

In this case, page replacement does not require reading from disk and can be performed very quickly

This trick can improve efficiency if page-replacement algorithm is making bad choices

If a page is replaced right before it will be used, this can mitigate the performance penalty

These ideas can be used with one another and incorporated into the various replacement algorithms we have discussed