Ch9 Notes - 3/27/2023

Monday, March 27, 2023 10:54 AM

//
Lisa Jacklin
CS 321 Operating Systems
Lecture Notes 3/27/2023
//

Midterms are almost done being graded and hoping wednesday we will get them back

//
Chapter 9: Memory Management -> Virtual Memory

demand paging is our main focus today: demaid paging is when an OS is demanding memory

page vs frame:
- page is in the virtual memory
- frame is the physical memory

page fault
- if there is a reference to a page, first reference to that page will trap to OS
- when the OS requests a page that does not exist
- when a process tries to find a page that is not in memory and must be swapped in from physical memory to be used without fault

->THIS IS IMPORTANT FOR QUIZ AND OR EXAM!<-
Steps in handling a page fault
1. reference page table,
2. trap OS
3. page is on the backing store
...
->the rest of the steps are on slide 16 of ch 9

performance of demanding paging
- the demand paging, the more time spend waiting for the operation to complete

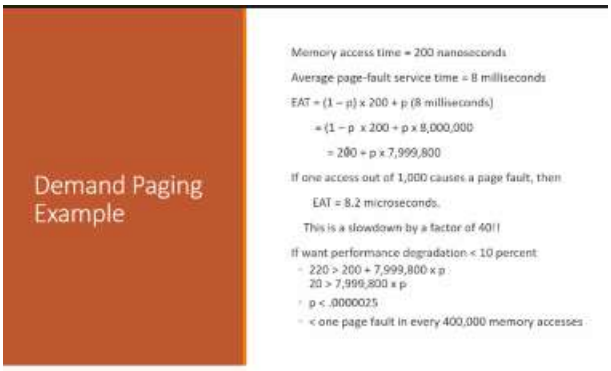
stages:
- trap to the OS
- save the user registers and process state
-determine that the interrupt was a page fault
- check that the page reference was legal and determine the location of the page on the memory
- issue a read from the disk to a free frame
- while waiting, allocate the CPU to some other
... more on slide 19...

three major activities
- service the interrupt : far less time than reading the page
- read the page : takes a larger amount of time to read that data
- restart the process: only a small amount of time

page fault rate 0 =< p =< 1
- if p = 0 no page faults
- if p = 1 every reference is at fault

Effective Access Time (EAT)

EAT = (1-p) x memory access + p (page fault overhead + swap page out + swap page in)



-> I would go through and make sure you understand the EAT equation!

- you can check your understanding with example on slide 21
- note that the average page-fault service time can include any of the three major activities listed above.

-> END OF MAJOR IMPORTANCE ON TESTS/QUIZES <-

Demand Paging Optimization (design considerations)
-> no questions on test over this
-> note that this uses SATA technology for some amount of time and now, faster….?

- Swap space I/O faster than file system I/O even if on the same device
  - Swap allocates in larger chunks, less management needed than file system
- Copy entire process image to swap space at process load time
  - Then page in and out of the swap space which is used primarily in older Unix systems
- Demand page in from program binary on disk but discard rather than paging out when freeing frame
  - Still need to write to swap space
  - Page not associated with a file (like stack and heap) - anonymous memory
  - Pages modified in memory but note yet written back to the file system
- Mobile systems typically don't support
- swapping

Copy on Write

- Copy-on-Write: allows both parent and child processes to initially share the same pages in memory
  - If either process modifies a shared page, only then is the page copied.
- COW allows more efficient process creation as only modified pages are copied.
- In general, free pages are allocated from a pool of zero fill on demand pages
  - Pool should always have free frames for fast demand page execution
    - Don’t want to have to free a frame as well as other processes on page fault
- Vfork() variation on fork() system acll has parent suspend and child using copy-on-write address space of parent
  - Designed to have child call exec()

- Very efficient

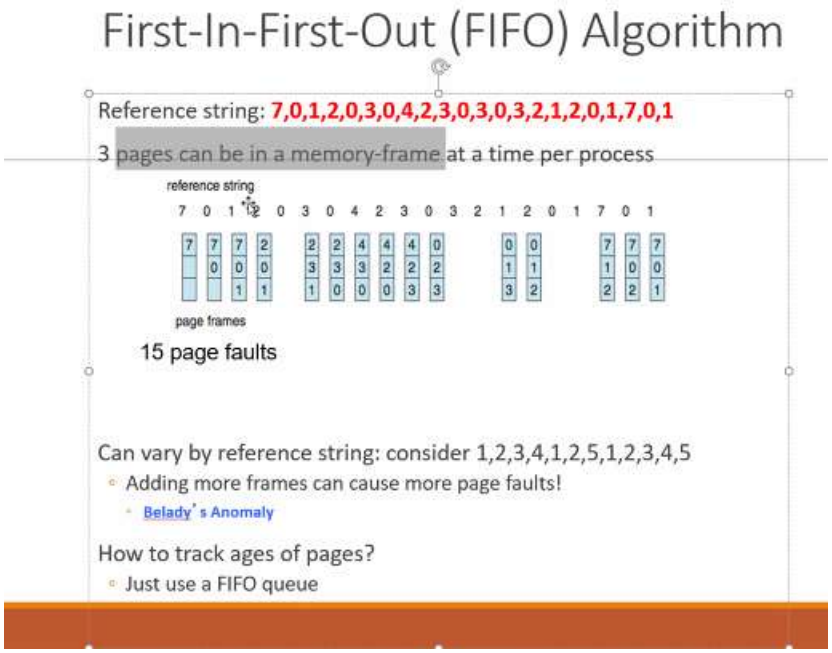
What happens if there is no free frame?
- Used up by process pages
- Also in demand from the kernel, I/O buffers, etc
- Page replacement: find some page in memory, but not really in use, page it out
  - Algorithm: terminate, swap out, replace?
  - Performance: want an algorithm which will result in minimum number of page faults

Page replacement
- Prevent over-allocation of memory by modifying page-fault service routine to include page replacement
- Use modify (dirty) bit to reduce overhead of page transfers only modified pages are written to disk
- Page replacement completes separation between logical memory and physical memory; large virtual memory can be provided on smaller memory

->view slide 29
Basic page replacement
- Find the location of the desired page on the disk
- Find a free frame:
  - If there is one, use it
  - Algorithm selects the victim frame if there is no free frame
- Bring the desired page into the newly free frame; update the page and frame table
- Continue the process by restarting the instruction that caused the trap

…

Page and frame replacement algorithms
-> slide 31
- Primarily using reference string algorithm…



- The larger the size of the frame, the more page faults, the larger the number of frames, the less amount of page faults

-> 32, 33, 34 are all visual and count help with understanding if you take a look at them and take a

moment to understand them.. Have few notes but can explain if necessary ;)

Ended lecture on slide 35.