Lecture - XVI
Virtual Memory - II

Tevfik Koşar

University at Buffalo
October 25th, 2012

Roadmap

- Virtual Memory
  - Page Replacement Algorithms
    - Optimal Algorithm
    - Least Recently Used (LRU)
    - LRU Approximations
    - Counting Algorithms
  - Allocation Policies
  - Thrashing
  - Working Set Model

FIFO

- FIFO is obvious, and simple to implement
  - when you page in something, put it on the tail of a list
  - evict page at the head of the list
- Why might this be good?
  - maybe the one brought in longest ago is not being used
- Why might this be bad?
  - then again, maybe it is being used
  - have absolutely no information either way
- In fact, FIFO’s performance is typically lousy
- In addition, FIFO suffers from Belady’s Anomaly
  - there are reference strings for which the fault rate increases when the process is given more physical memory

Optimal Algorithm

- Replace page that will not be used for the longest time in future
- 4 frames example
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Optimal (Belady’s) Algorithm

- Provably optimal: lowest fault rate (remember SJF?)
  - evict the page that won’t be used for the longest time in future
  - problem: impossible to predict the future
- Why is Belady’s Optimal algorithm useful?
  - as a yardstick to compare other algorithms to optimal
    - if Belady’s isn’t much better than yours, yours is pretty good
    - how could you do this comparison?
- Is there a best practical algorithm?
  - no, depends on workload
- Is there a worst algorithm?
  - no, but random replacement does pretty badly
    - there are some other situations where OS’s use near-random algorithms quite effectively!
Least Recently Used (LRU)

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

LRU uses reference information to make a more informed replacement decision
- Idea: past experience gives us a guess of future behavior
- On replacement, evict the page that hasn’t been used for the longest amount of time
  - LRU looks at the past, Belady’s wants to look at future
  - How is LRU different from FIFO?

Implementation
- To be perfect, must grab a timestamp on every memory reference, then order or search based on the timestamps...
- Way too costly in memory bandwidth, algorithm execution time, etc.
- So, we need a cheap approximation...

LRU Implementations

- Stack implementation - keep a stack of page numbers in a double link form:
  - Page referenced: move it to the top
  - Requires 6 pointers to be changed
  - No search for replacement

LRU Approximation Algorithms

- Reference bit
  - With each page associate a bit, initially = 0
  - When page is referenced bit set to 1
  - Replace the one which is 0 (if one exists). We do not know the order, however.

- Additional Reference bits
  - 1 byte for each page: eg. 00110011
  - Shift right at each time interval

LRU Clock Algorithm

- AKA Not Recently Used (NRU) or Second Chance
  - Replace page that is “old enough”
  - Logically, arrange all physical page frames in a big circle (clock)
    - Just a circular linked list
    - A “clock hand” is used to select a good LRU candidate
    - Sweep through the pages in circular order like a clock
    - If ref bit is off, it hasn’t been used recently, we have a victim
    - So, what is minimum “age” if ref bit is off?
    - If the ref bit is on, turn it off and go to next page
  - Arm moves quickly when pages are needed
  - Low overhead if have plenty of memory
  - If memory is large, “accuracy” of information degrades
    • Add more hands to fix
Second-Chance (clock) Page-Replacement Algorithm

Counting Algorithms

- Keep a counter of the number of references that have been made to each page
- LFU Algorithm: replaces page with smallest count
- MFU Algorithm: based on the argument that the page with the smallest count was probably just brought in and has yet to be used

Allocation of Frames

- Each process needs *minimum* number of pages
- Two major allocation schemes
  - fixed allocation
  - priority allocation

Fixed Allocation

- Equal allocation - For example, if there are 100 frames and 5 processes, give each process 20 frames.
- Proportional allocation - Allocate according to the size of process

Priority Allocation

- Use a proportional allocation scheme using priorities rather than size
- If process \( P_i \) generates a page fault,
  - select for replacement one of its frames
  - select for replacement a frame from a process with lower priority number

Global vs. Local Allocation

- Global replacement - process selects a replacement frame from the set of all frames; one process can take a frame from another
- Local replacement - each process selects from only its own set of allocated frames
Thrashing

- If a process does not have "enough" frames, the page-fault rate is very high. This leads to:
  - Replacement of active pages which will be needed soon again
    - **Thrashing** = a process is busy swapping pages in and out
  - Which will in turn cause:
    - low CPU utilization
    - operating system thinks that it needs to increase the degree of multiprogramming
    - another process added to the system

Thrashing (Cont.)

Locality in a Memory-Reference Pattern

- $\Delta = \text{working-set window} = \text{a fixed number of page references}$
  - Example: 10,000 instruction
- $WSS_i (\text{working set of Process } P_i) =$ total number of pages referenced in the most recent $\Delta$ (varies in time)
  - if $\Delta$ too small will not encompass entire locality
  - if $\Delta$ too large will encompass several localities
  - if $\Delta = \infty$ will encompass entire program
- $D = \Sigma WSS_i =$ total demand frames
  - if $D > m$ $\Rightarrow$ Thrashing
  - Policy if $D > m$, then suspend one of the processes

Working-set model

- Consider the following page-reference string:
  - 1, 2, 3, 4, 4, 3, 2, 1, 5, 6, 2, 1, 2, 3, 7, 8, 3, 2, 1, 5
- Assuming 4 memory frames and LFU, LRU, or Optimal page replacement algorithms, how many page faults, page hits, and page replacements would occur? Show your page assignments to frames.

Exercise
Summary

- Virtual Memory
  - Page Replacement Algorithms
  - Optimal Algorithm
  - Least Recently Used (LRU)
  - LRU Approximations
  - Counting Algorithms
  - Allocation Policies
  - Thrashing
  - Working Set Model

- **Next Lecture: Project 2 Discussion**
- **Reading Assignment: Chapter 9 from Silberschatz.**

Acknowledgements

- “Modern Operating Systems” book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from UNR
- Gribble, Lazowska, Levy, and Zahorjan from UW