

More on Virtual Memory

CS 256/456

Dept. of Computer Science, University of Rochester

Recap of the Last Class

- **Virtual memory** - separation of user logical memory from physical memory.
 - Transparent page sharing:
 - Copy-on-write: allows for more efficient process creation.
 - Only part of the program address space needs to be in physical memory for execution:
 - Demand paging.
 - Memory-mapped I/O.
- Page replacement algorithm: the algorithm that picks the victim page.
 - FIFO, Optimal, LRU.

Implementations of Page Replacement Algorithms

- FIFO implementation.
- LRU implementations:
 - Time-of-use implementation
 - Stack implementation
- What needs to be done at each memory reference?
- What needs to be done at page loading or page replacement?

LRU Approximation Algorithms

- LRU approximation with a little help from the hardware.
- Reference bit
 - With each page associate a bit, initially = 0
 - When page is referenced, the bit is set to 1 by the hardware.
 - Replace a page whose reference bit is 0 (if one exists). We do not know the order, however.
- Second chance
 - Combining the reference bit with FIFO replacement
 - If page to be replaced (in FIFO order) has reference bit = 1, then:
 - set reference bit 0.
 - leave page in memory.
 - replace next page (in FIFO order), subject to same rules.

LRU Approximation Algorithms

- Enhancing the reference bit algorithm:
 - it would be nice if there is more information about the reference history than a single bit.
 - with some help from software.
- Maintain more reference bits in software:
 - at every N-th clock interrupt, the OS moves each hardware page reference bit into a page reference history word (with more than one bit).

2/27/2006

CSC 256/456 - Spring 2006

5

Counting-based Page Replacement

- Least frequently used page-replacement algorithm
 - the page with smallest access count (within a period of time) is replaced
- Implementation?

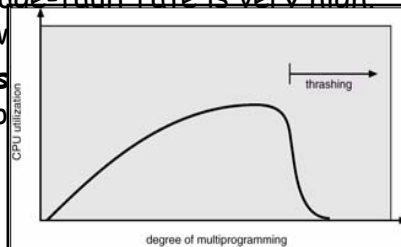
2/27/2006

CSC 256/456 - Spring 2006

6

Per-process Memory Allocation and Thrashing

- Our discussion of memory replacement is at global level, with no concern of how much memory each process gets
- If a process does not have "enough" pages, the page-fault rate is very high. This leads to low
- Thrashing with swapp

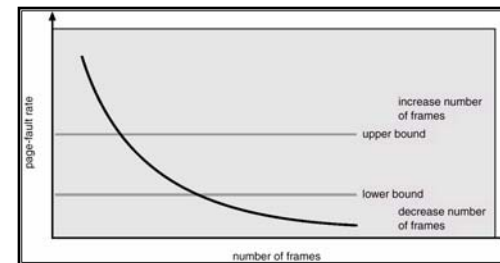


2/27/2006

CSC 256/456 - Spring 2006

7

Page-Fault Frequency Scheme



- Establish "acceptable" page-fault rate range.
 - If actual rate too low, process loses frame.
 - If actual rate too high, process gains frame.

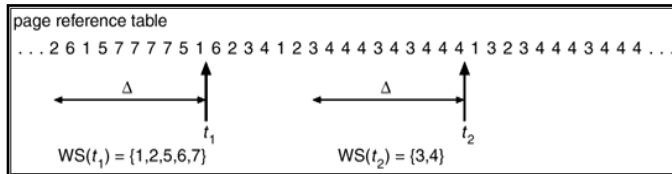
2/27/2006

CSC 256/456 - Spring 2006

8

Working-Set Model

- data access locality.
- WSS_i (working set of Process P_i) = total number of pages referenced in the most recent Δ (working-set window).



- Policy:
 - try to allocate enough frames for each process's working set.
 - if $\sum WSS_i > m$, then suspend one of the processes.

Transparent Sharing of Memory Pages

- You are asked to implement a memory management system that transparently discover and share pages of the same content over multiple processes.
 - pay some overhead is fine, but not too excessive.
 - discover: compare hash coding of pages.
 - share: copy-on-write.
- How often do pages have the same content?

Other Memory Management Issues

- When to swap out pages?
- Prepaging
 - swap in pages that are expected to be accessed in the future
- Page size selection
 - fragmentation
 - page table size
 - TLB reach

TLB Reach

- TLB Reach** - the amount of memory accessible from the TLB.
 - TLB Reach = (TLB Size) X (Page Size)
- Advantage of a large TLB reach?
- Increasing the TLB reach:
 - Increase the Page Size.** This may lead to an increase in fragmentation as not all applications require a large page size.
 - Provide Multiple Page Sizes.** This allows applications that require larger page sizes the opportunity to use them without an increase in fragmentation.

User Program Optimization

- Assume:
 - page size is 4KB
 - the program is allocated 1023 physical frames.
- Program structure
 - `int A[][] = new int[1024][1024];`
 - Program 1
 - `for (j = 0; j < A.length; j++)`
 - `for (i = 0; i < A.length; i++)`
 - `A[i,j] = 0;`
 - 1024 x 1024 page faults
 - Program 2
 - `for (i = 0; i < A.length; i++)`
 - `for (j = 0; j < A.length; j++)`
 - `A[i,j] = 0;`
 - 1024 page faults

Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).