CSC 4103 - Operating Systems
Spring 2008

LECTURE - XVII

VIRTUAL MEMORY - III

Tevfik Koşar

Louisiana State University

April 3rd, 2008

Performance of Demand Paging

- Page Fault Rate $0 \le p \le 1.0$
  - if $p = 0$ no page faults
  - if $p = 1$, every reference is a fault
- Effective Access Time (EAT)

EAT = (1 – p) x memory access
+ p x (page fault overhead
+ [swap page out]
+ swap page in
+ restart overhead)

Demand Paging Example

- Memory access time = 1 microsecond
- 50% of the time the page that is being replaced has been modified and therefore needs to be swapped out
- Swap Page Time = 10 msec = 10,000 microsec
- EAT = (1 – p) x 1 + p x (10,000 + 1/2 x 10,000)
  = 1 + 14,999 x p (in microsec)
- What if 1 out of 1000 memory accesses cause a page fault?
- What if we only want 30% performance degradation?

Exercise

- Assume we have a demand-paged memory, with memory access time 1 milliseconds. It takes 10 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Assume that the page to be replaced is not modified 60 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 5 milliseconds?

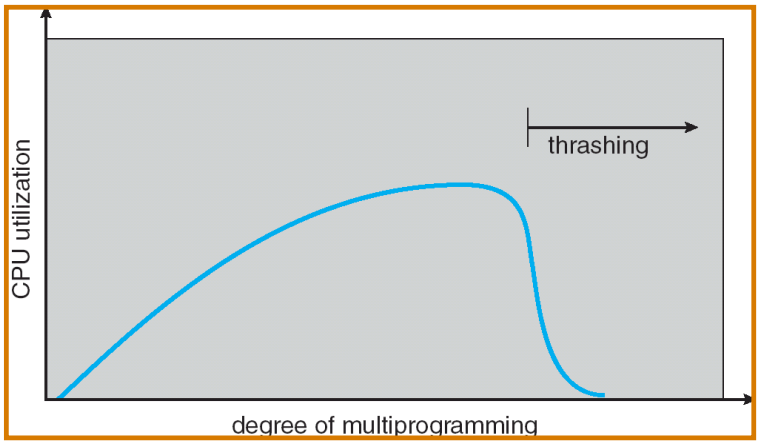
Solution

- 5 > (1-p)*1 + p*(0.6*10+0.4*20)
- 5 > 1-p+14p
- 5 > 1-13p
- p < 4/13
- p < 0.30

Thrashing

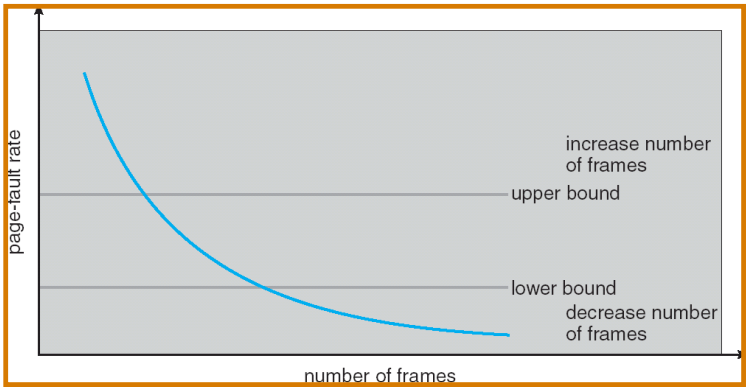
- If a process does not have "enough" frames, the page-fault rate is very high. This leads to:
  - Replacement of active pages which will be needed soon again
  - ➔ **Thrashing** ≡ a process is busy swapping pages in and out
- Which will in turn cause:
  - low CPU utilization
  - operating system thinks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

Thrashing (Cont.)



Page-Fault Frequency Scheme

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



Allocation of Frames

- Each process needs *minimum* number of pages
- Example: IBM 370 – 6 pages to handle SS MOVE instruction:
  - instruction is 6 bytes, might span 2 pages
  - 2 pages to handle *from*
  - 2 pages to handle *to*
- 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

s_i = size of process p_i

$S = \sum s_i$

m = total number of frames

a_i = allocation for $p_i = \frac{s_i}{S} \times m$

$m = 64$

$s_i = 10$

$s_2 = 127$

$a_1 = \frac{10}{137} \times 64 \approx 5$

$a_2 = \frac{127}{137} \times 64 \approx 59$

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 Replacement

- **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

Acknowledgements

- “Operating Systems Concepts” book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne
- “Operating Systems: Internals and Design Principles” book and supplementary material by W. Stallings
- “Modern Operating Systems” book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from UNR