# Virtual Memory (4)

Dr. Jun Zheng

CSE325 Principles of Operating Systems

11/11/2019

# In-Class Work 6

Consider the following page reference string:

`1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6`

How many page faults would occur for the following replacement algorithms, assuming three frames?

Remember that all frames are initially empty, so your first unique pages will cost one fault each.

- ❑ LRU replacement
- ❑ FIFO replacement
- ❑ Optimal replacement

# LRU Approximation Algorithms

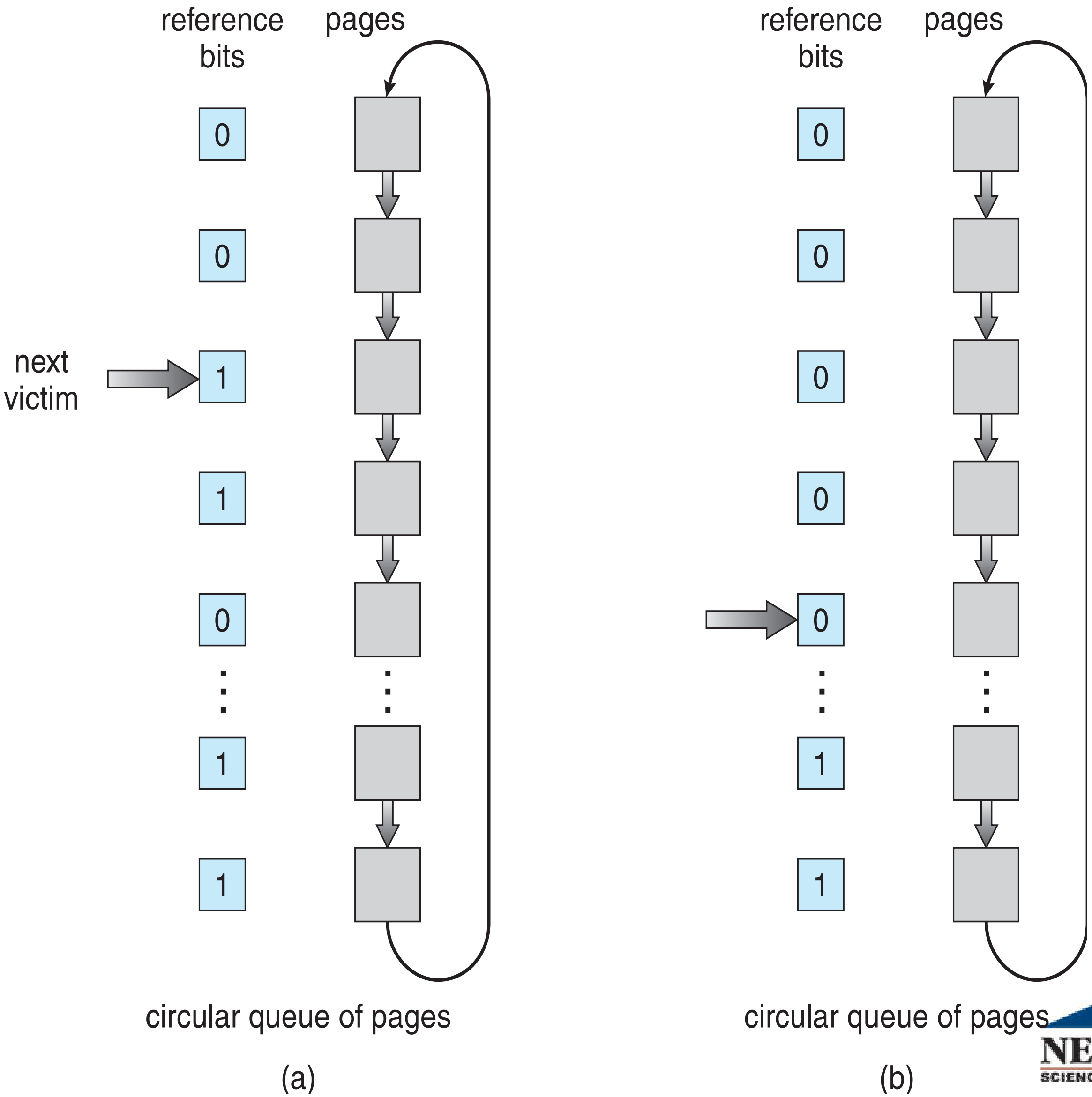
- LRU needs special hardware and still slow
- **Reference bit**
  - With each page associate a bit, initially = 0
  - When page is referenced bit set to 1
  - Replace any with reference bit = 0 (if one exists)
    - We do not know the order, however
  - Basis for many LRU approximation algorithms

# LRU Approximation Algorithms

- **Second-chance algorithm**
  - Generally FIFO, plus hardware-provided reference bit
  - If page to be replaced has
    - Reference bit = 0 -> replace it
    - reference bit = 1 then:
      - set reference bit 0, leave page in memory
      - replace next page, subject to same rules

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

# Second-Chance (clock) Page-Replacement Algorithm



# Enhanced Second-Chance Algorithm

- Improve algorithm by using reference bit and modify bit (if available)
- Take ordered pair (reference, modify)
  - (0, 0) neither recently used not modified – best page to replace
  - (0, 1) not recently used but modified – not quite as good, must write out before replacement
  - (1, 0) recently used but clean – probably will be used again soon
  - (1, 1) recently used and modified – probably will be used again soon and need to write out before replacement
- When page replacement called for, use the clock scheme but use the four classes replace page in lowest non-empty class
  - Might need to search circular queue several times

# Frame Allocation

- Each process needs minimum number of frames
  - E.g. IBM 370 – 6 pages to handle MVC instruction
- Maximum that a process can ge of course is total frames in the system
- Two major allocation schemes
  - Fixed allocation
  - Priority allocation
- Many variations

# Fixed Allocation

- Equal allocation
  - E.g. if there are 100 frames (after allocating frames for the OS) and 5 processes, give each process 20 frames
- Proportional allocation – allocate according to the size of process
  - Dynamic as degree of multiprogramming, process sizes change

$$s_i = \text{size of process } p_i$$

$$S = \sum s_i$$

$$m = \text{total number of frames}$$

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

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

# 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

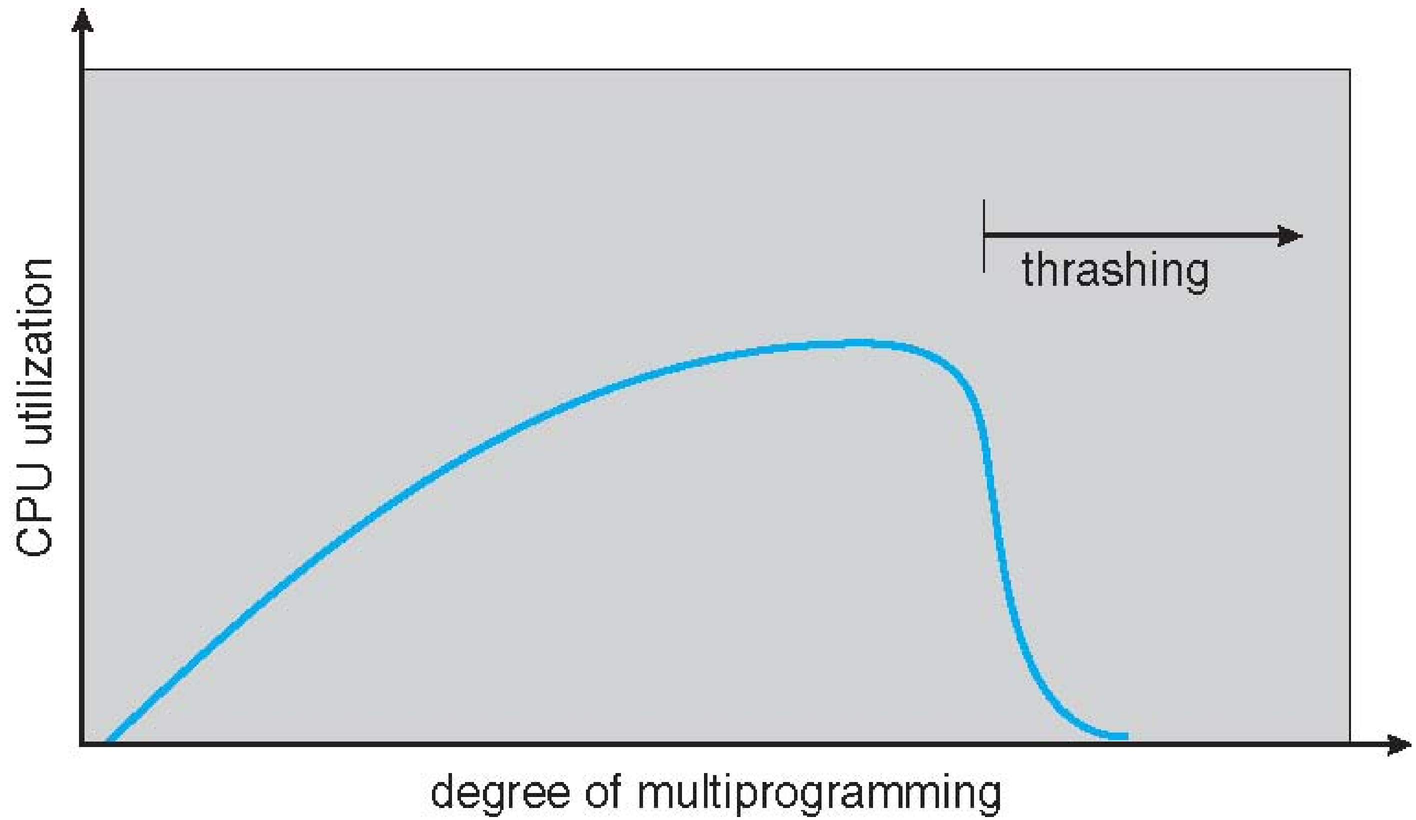
# Global vs. Local Allocation

- **Global replacement** – process selects a replacement frame from the set of all frames; one process can take a frame from another
  - A process cannot control its own page-fault rate
  - Greater throughput so more common
- **Local replacement** – each process selects from only its own set of allocated frames
  - More consistent per-process performance
  - But possibly underutilized memory

# Thrashing

- If a process does not have “enough” pages, the page-fault rate is very high
  - Page fault to get page
  - Replace existing frame
  - But quickly need replaced frame back
  - This leads to:
    - Low CPU utilization
    - Operating system thinking that it needs to increase the degree of multiprogramming
    - Another process added to the system
- **Thrashing** ≡ a process is busy swapping pages in and out

# Thrashing (Cont.)

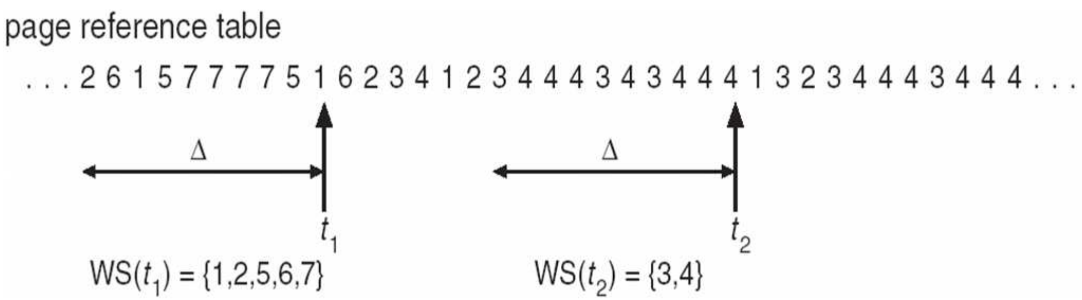


# Demand Paging and Thrashing

- Why does demand paging work?
  **Locality model**
  - Process migrates from one locality to another
  - Localities may overlap
- Why does thrashing occur?
  $\Sigma$ size of locality > total memory size
  - Limit effects by using local or priority page replacement

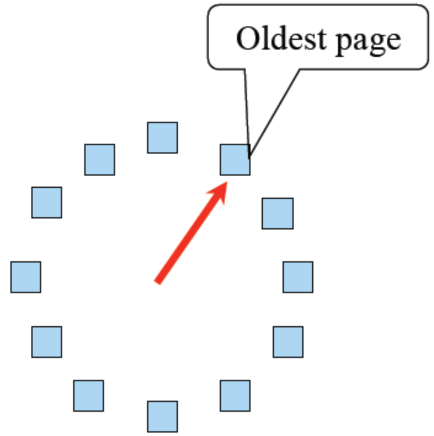
# Working Set

- Main idea
  - Define a working set as the set of pages in the most recent K page references to approximate the program's locality
  - Keep the working set in memory will reduce page faults significantly
- Approximate working set
  - The set of pages of a process used in the last T seconds



# WSClock

- Follow the clock hand
- If the reference bit is 1
  - Set reference bit to 0
  - Set the current time for the page
  - Advance the clock hand
- If the reference bit is 0, check "time of last use"
  - If the page has been used within δ, go to the next
  - If the page has not been used within δ and modify bit is 1
    - Schedule the page for page out and go to the next
  - If the page has not been used within δ and modify bit is 0
    - Replace this page




"WSCLOCK—a simple and effective algorithm for virtual memory management" by Richard W. Carr and John L. Hennessy, SOSP'81.

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY