CSE 421/521 - Operating Systems
Fall 2018

LECTURE - XVII

VIRTUAL MEMORY - II

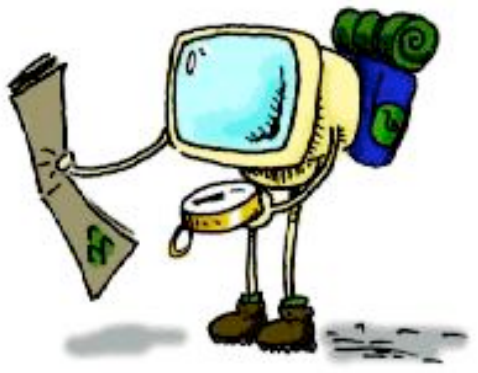
Tevfik Koşar

University at Buffalo

October 25th, 2018

Roadmap

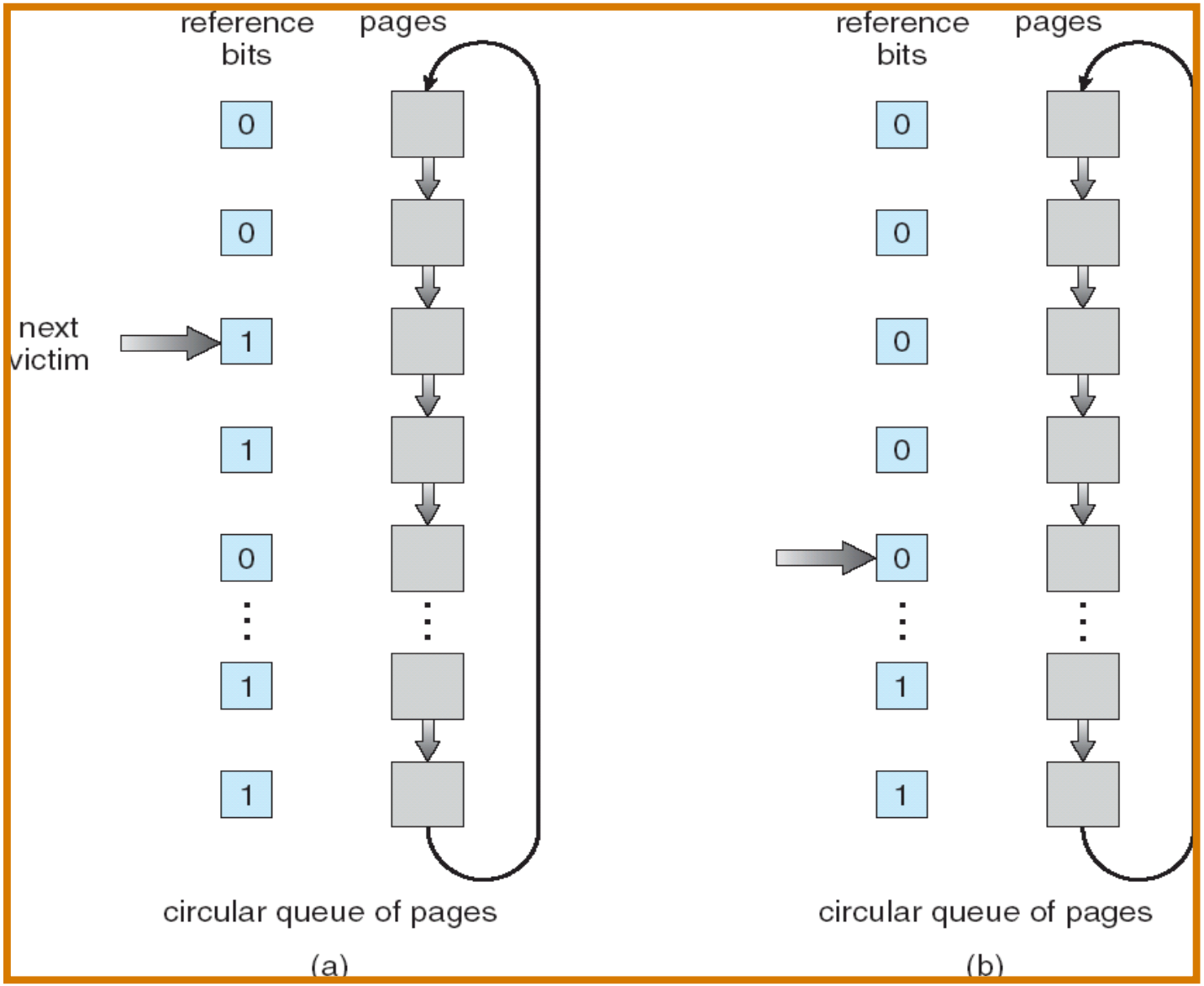
- Virtual Memory
  - LRU Clock Algorithm
  - Counting Algorithms
  - Performance of Demand Paging
  - Allocation Policies
  - Thrashing
  - Working Set Model



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



Example

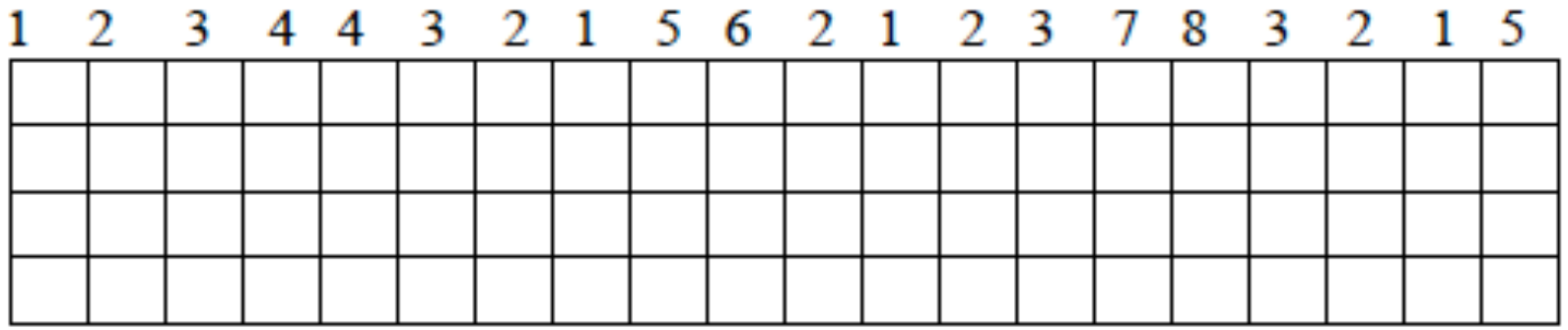
- Consider the following page-reference string:

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 LRU-Clock algorithm, how many page faults, page hits, and page replacements would occur? Show your page assignments to frames. .

Consider the following rules:

1. When a page is brought to the memory the first time, initialize reference bit to 0.
2. Advance the next victim pointer only if you need to find a victim page to replace, and when you bring a new page in.



# of page faults:
# of page hits:
# of page replacements:

Counting Algorithms

- Keep a counter of the number of references that have been made to each page

- **LFU Algorithm**: replaces page with smallest count

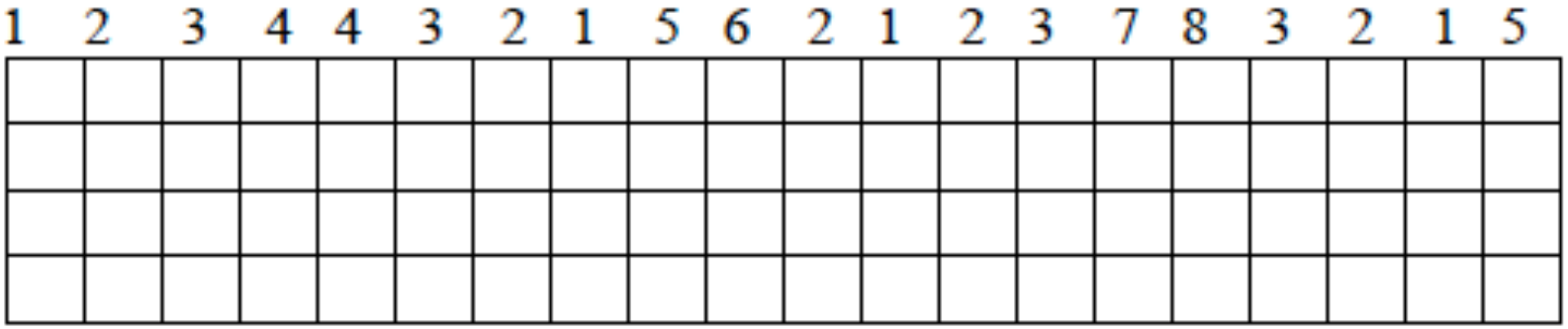
- **MFU Algorithm**: based on the argument that the page with the smallest count was probably just brought in and has yet to be used

Exercise

- 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 LRU, LRU-8bit, LFU, MFU, or Optimal page replacement algorithms, how many page faults, page hits, and page replacements would occur? Show your page assignments to frames.



# of page faults:
# of page hits:
# of page replacements:

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)

$$\begin{aligned} EAT = {} & (1 - p) \times \text{memory access} \\ & + p \times (\text{page fault overhead} \\ & \quad + [\text{swap page out}] \\ & \quad + \text{swap page in} \\ & \quad + \text{restart overhead} \\ & \quad + \text{memory access}) \end{aligned}$$

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 = ?

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)
  = 1 + 15,000 x p (in microsec)
- What if 1 out of 1000 memory accesses cause a page fault?
- What if we only want 30% performance degradation?

Question - 1

Consider a paging system with the page table stored in memory.

a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

b) If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.

Question - 1

Consider a paging system with the page table stored in memory.

a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

==> 400 nanoseconds; 200 nanoseconds to access the page table and 200 nanoseconds to access the word in memory.

b) If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.

==> Effective access time = 0.75 x (200 nanoseconds) + 0.25 x (400 nanoseconds) = 250 nanoseconds.

Question 2 (a)

Consider a demand paging system where 40% of the page table is stored in the registers, and the rest needs to be addressed from the memory. Assume any reference to the memory takes 100 nanoseconds, and 20% of the time the page that is being replaced has its dirty bit set to 1. Swapping a page in takes 1000 microseconds and swapping out a page takes 2000 microseconds.

(a) How long does a paged memory reference take?

Question 2 (a)

Consider a demand paging system where 40% of the page table is stored in the registers, and the rest needs to be addressed from the memory. Assume any reference to the memory takes 100 nanoseconds, and 20% of the time the page that is being replaced has its dirty bit set to 1. Swapping a page in takes 1000 microseconds and swapping out a page takes 2000 microseconds.

(a) How long does a paged memory reference take?

A paged memory reference would take (60/100) x 100 + 100 = 160 ns

Question 2 (b)

(b) What is the Effective Access Time (EAT) if the page fault ratio is 0.001?

Question 2 (b)

(b) What is the Effective Access Time (EAT) if the page fault ratio is 0.001?

$$EAT = (1-p) \times 0.160 + p \times ((80/100) \times 1000 + (20/100) \times 3000 + 0.160)$$

$$= 0.160 + 1400p$$

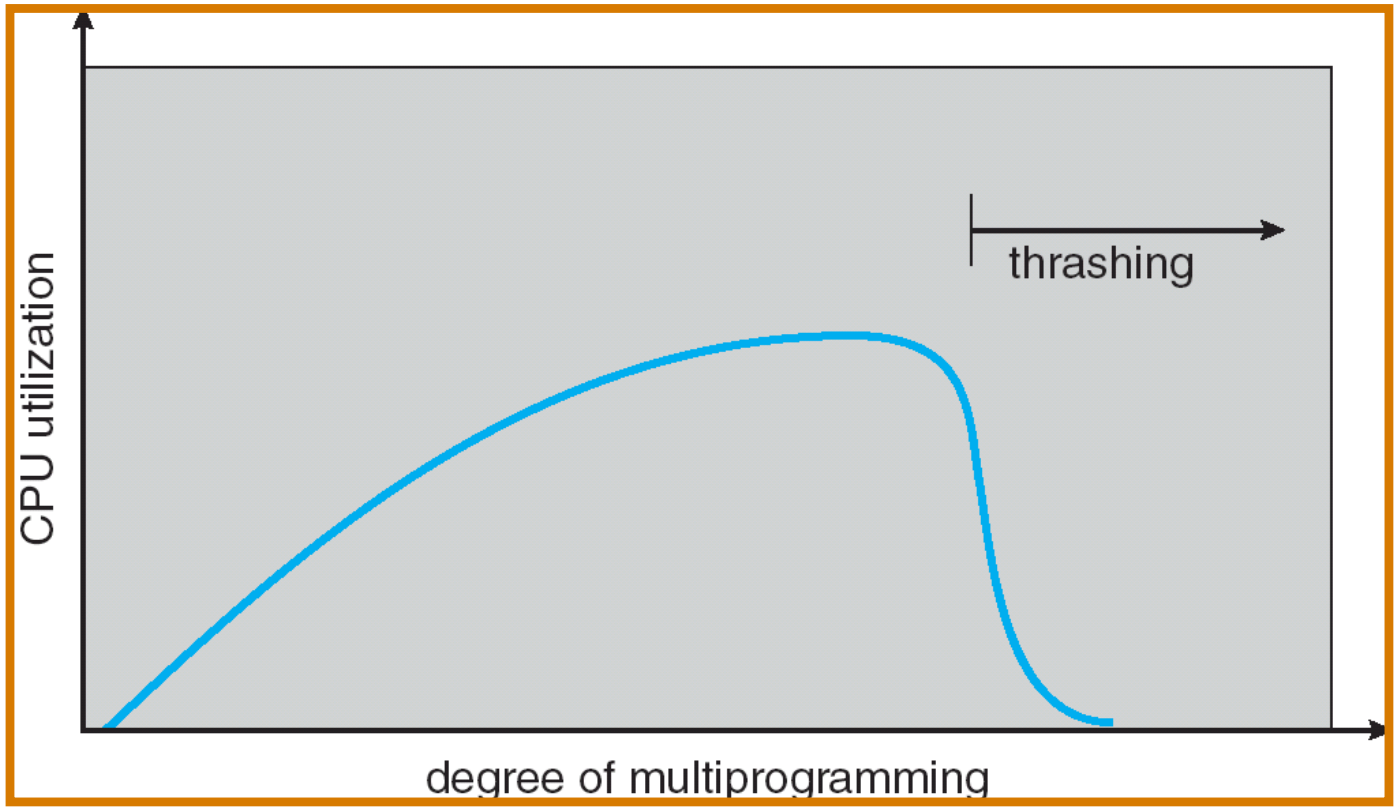
$$= 0.160 + 1400 \times 1/1000$$

$$= 1.56 \text{ microseconds}$$

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.)



Exercise - 1 (a)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

a) CPU utilization 96 percent; disk utilization 4 percent.

Exercise - 1 (a)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

a) CPU utilization 96 percent; disk utilization 4 percent.

Answer: CPU utilization is sufficiently high to leave things alone (there are already sufficient processes running to keep the CPU busy); increasing the degree of multiprogramming may decrease the CPU utilization.

Exercise - 1 (b)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

b) CPU utilization 10 percent; disk utilization 95 percent.

Exercise - 1 (b)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

b) CPU utilization 10 percent; disk utilization 95 percent.

Answer: Thrashing is occurring. We cannot increase the CPU utilization

Exercise - 1(c)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

c) CPU utilization 12 percent; disk utilization 2 percent.

Exercise - 1 (c)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

c) CPU utilization 12 percent; disk utilization 2 percent.

Answer: both CPU and disk utilization are low, and CPU is obviously underutilized. We should increase the degree of multiprogramming to increase CPU utilization.

Exercise - 2 (a)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(a) Install a faster CPU.

Exercise - 2 (a)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(a) Install a faster CPU.

NO. a faster CPU reduces the CPU utilization further since the CPU will spend more time waiting for a process to enter in the ready queue.

Exercise - 2 (b)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(b) Install a bigger paging disk.

Exercise - 2 (b)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(b) Install a bigger paging disk.

NO. the size of the paging disk does not affect the amount of memory that is needed to reduce the page faults.

Exercise - 2 (c)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(c) Decrease the degree of multiprogramming.

Exercise - 2 (c)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(c) Decrease the degree of multiprogramming.

YES. by suspending some of the processes, the other processes will have more frames in order to bring their pages in them, hence reducing the page faults.

Exercise - 2 (d)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(d) Install more main memory.

Exercise - 2 (d)

- Consider a demand-paging system with the following time-measured utilization:

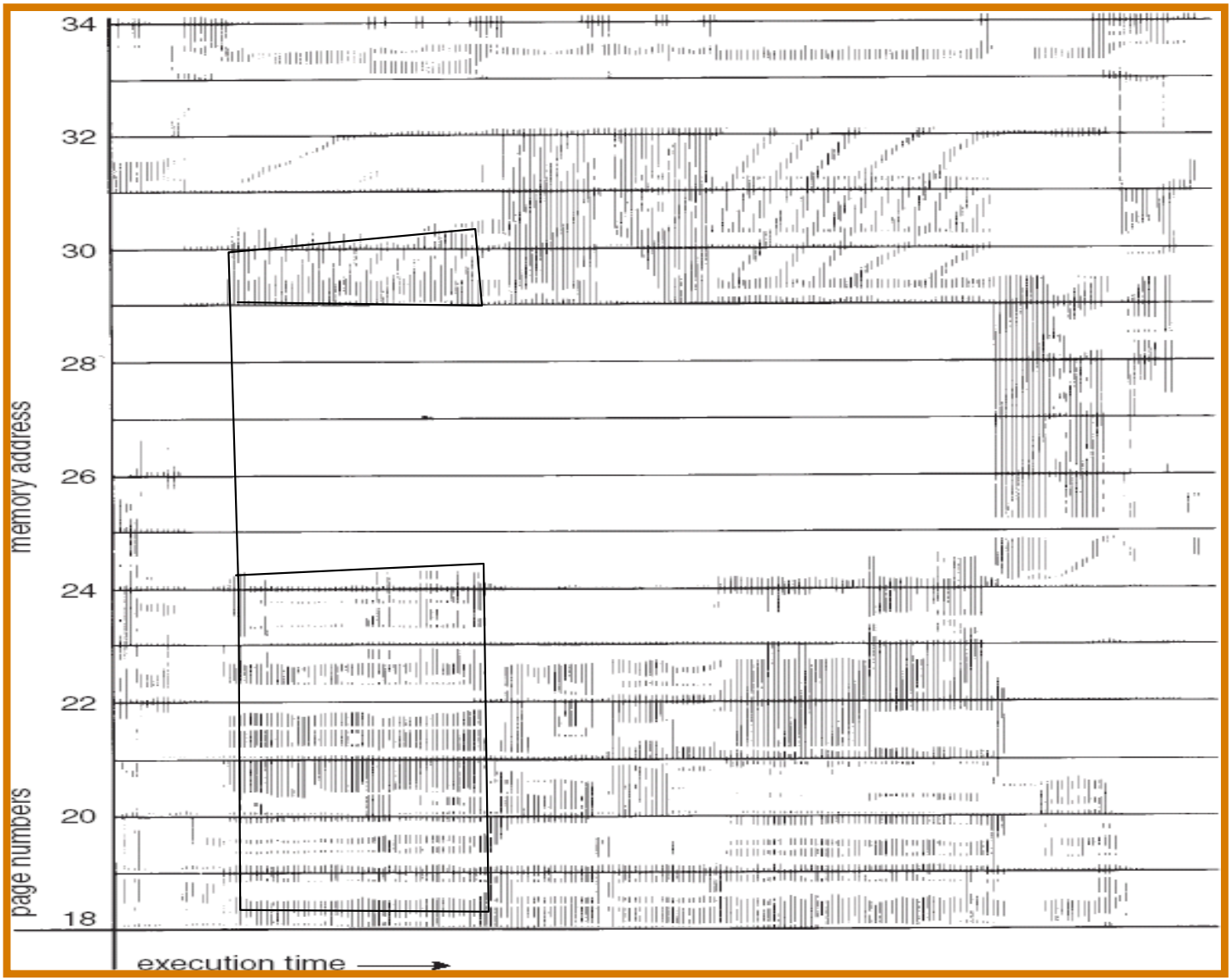
  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(d) Install more main memory.

Likely. more pages can remain resident and do not require paging to or from the disks (i.e. would depend on the page replacement algorithm you are using and the page reference sequence).

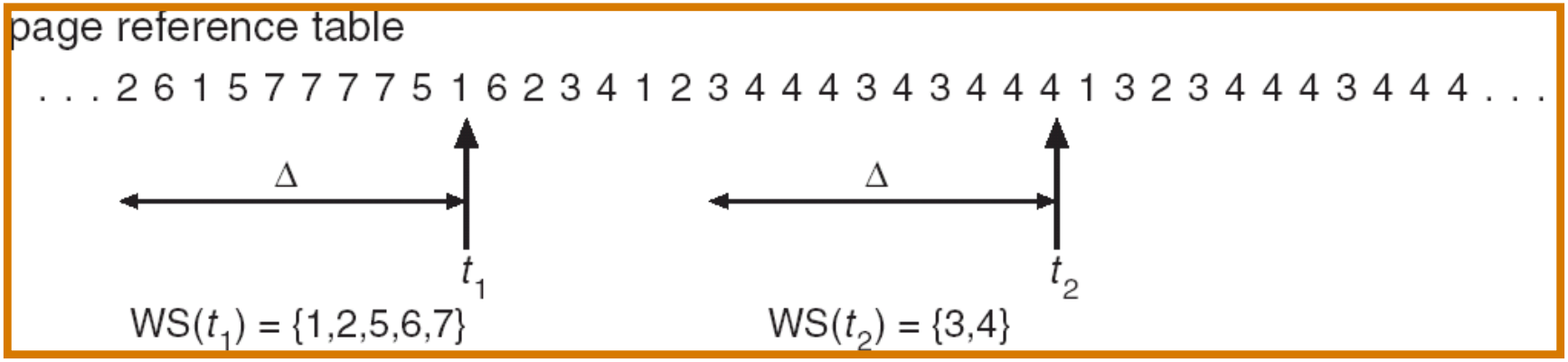
Locality in a Memory-Reference Pattern



Working-Set Model

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

Working-set model



Allocation of Frames

- We should allocate each process at least a *minimum* number of frames.

- 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

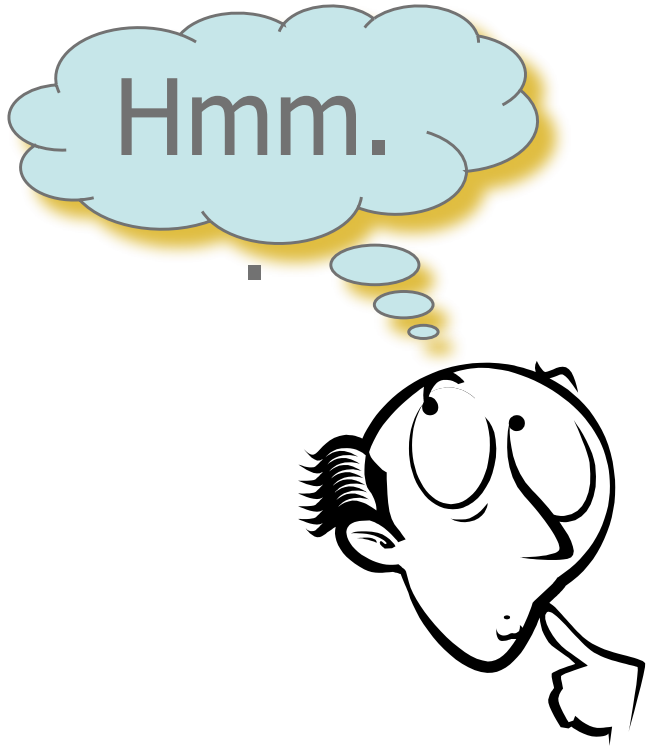
- If process P_i generates a page fault,
  - select for replacement one of its frames
  - select for replacement a frame from a process with lower priority number

Global vs. Local 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

Summary

- Virtual Memory
  - LRU-Clock Algorithm
  - Counting Algorithms
  - Performance of Demand Paging
  - Allocation Policies
  - Thrashing
  - Working Set Model



- Reading Assignment: Chapter 9 from Silberschatz.

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
- Gribble, Lazowska, Levy, and Zahorjan from UW