OPERATING SYSTEMS

Memory Management - 11

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 3

Unit-3:Unit 3: Memory Management: Main Memory

Hardware and control structures, OS support, Address translation, Swapping, Memory Allocation (Partitioning, relocation), Fragmentation, Segmentation, Paging, TLBs context switches

Virtual Memory - Demand Paging, Copy-on-Write, Page replacement policy - LRU (in comparison with FIFO & Optimal), Thrashing, design alternatives - inverted page tables, bigger pages.

Case Study: Linux/Windows Memory

Course Outline

25	Main Memory: Hardware and control structures, OS support, Address translation	8.1	64.2
26	Dynamic linking, Swapping	8.2	
27	Memory Allocation (Partitioning, relocation), Fragmentation	8.3	
28	Segmentation	8.4	
29	Paging: OS Support, TLBs, Address Translation	8.5	
30	Structure of the Page Table	8.6	
31	Design Alternatives – Inverted Page Tables, Bigger Pages	8.7-8.8	
32	Virtual Memory: Demand Paging, Copy-OnWrite	9.1-9.3	
33	Page replacement policy – LRU	9.4	
34	FIFO & Optimal	9.5	
35	Thrashing	9.6	
36	Case Study: Linux/ Windows Memory Management	9.10	

Topic Outline

- Virtual Memory - Page replacement
- What happens if there is no free Frame ?
- Basic Page Replacement
- Page and Frame Replacement Algorithms
- Graph of Page Faults versus the number of Frames

Topic Outline

- First-In-First-Out (FIFO) Algorithm
- FIFO illustrating Belady’s Anomaly
- Optimal Page Replacement Algorithm
- Least Recently Used (LRU) Algorithm
- Use of a Stack to Record Most Recent Page References

Topic Outline

- LRU Algorithm Implementation
- LRU Approximation Algorithm
- Second-Chance (clock) Page-Replacement Algorithm
- Enhanced Second-Chance Algorithm
- Counting Algorithms
- Page-Buffering Algorithms

Topic Outline

- Applications and Page Replacement
- Allocation of Frames
- Fixed Allocation
- Priority Allocation
- Global vs. Local Allocation
- Non-Uniform Memory Access

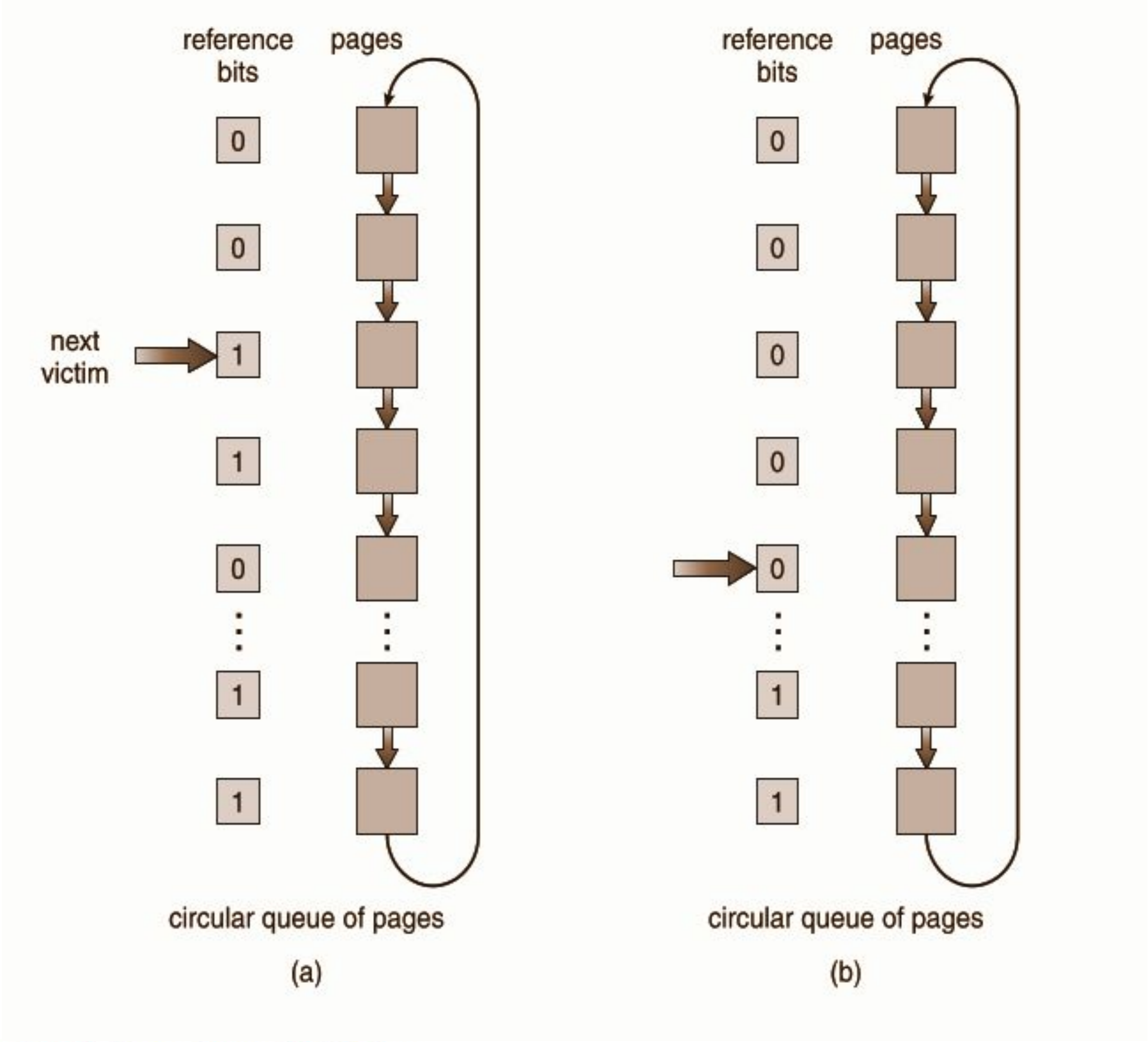
LRU Approximation Algorithms

- 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

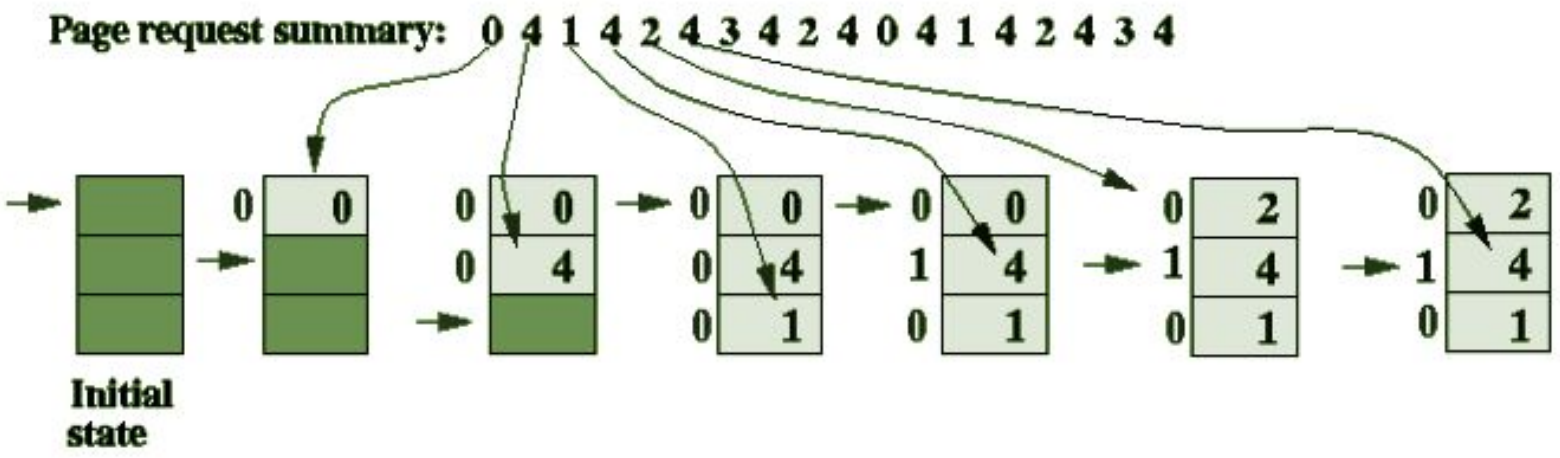
Second-Chance (clock) Page-Replacement Algorithm

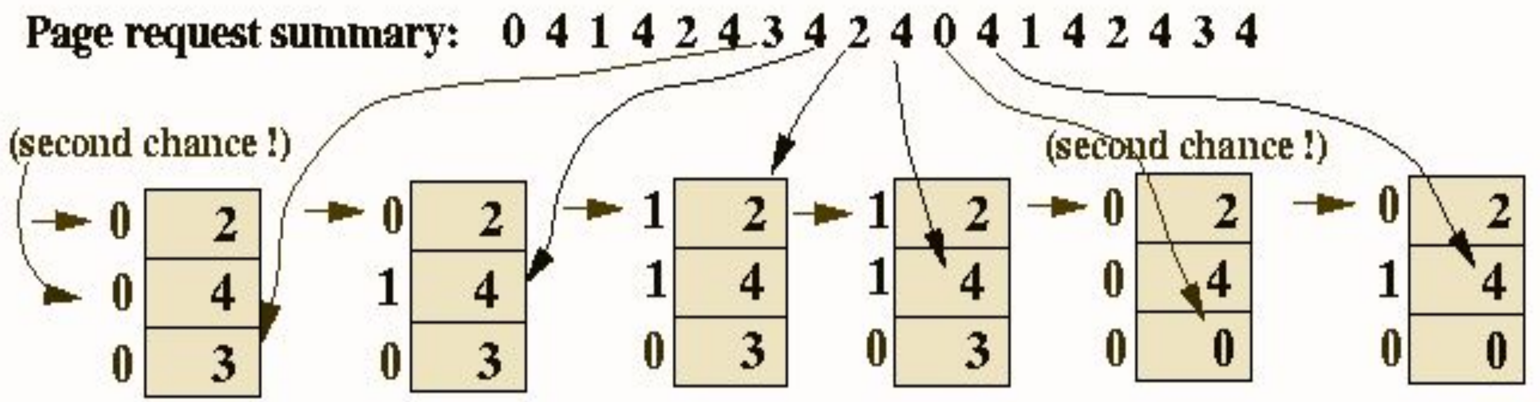
- Second-chance algorithm
  - Generally FIFO, plus hardware-provided reference bit
  - Clock replacement
    - If page to be replaced has reference bit = 0 -> replace it
    - if reference bit = 1 then:
      - set reference bit 0, leave page in memory
      - replace next page, subject to same rules

Second-Chance (clock) Page-Replacement Algorithm

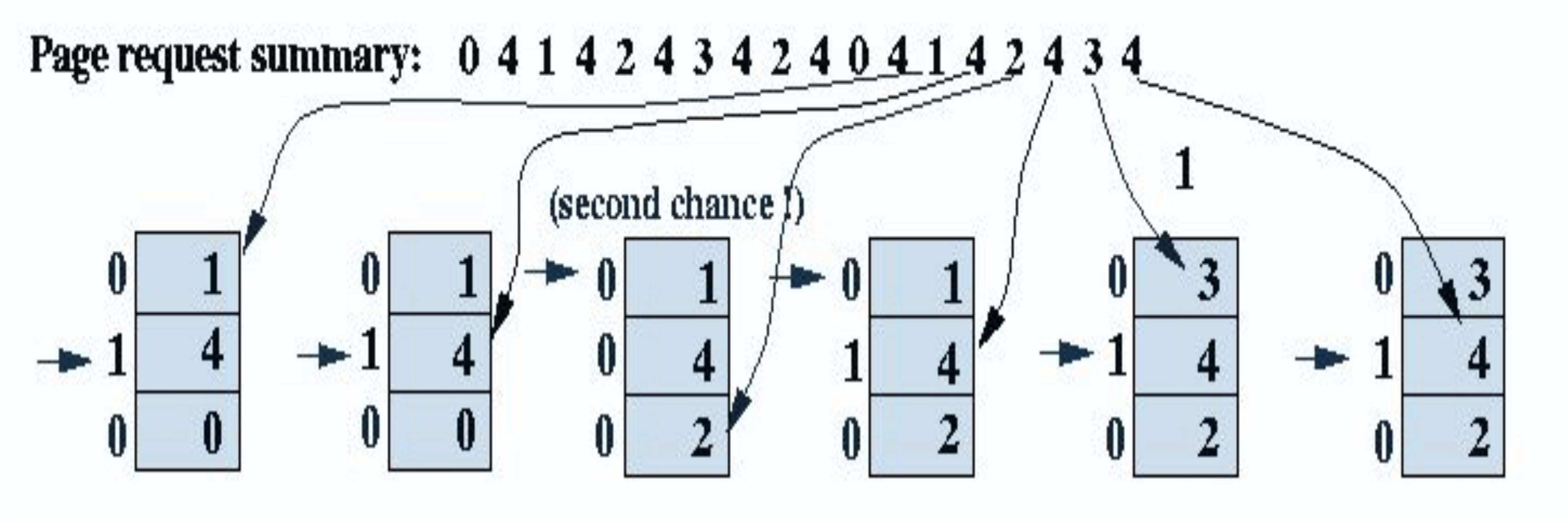


Second-Chance (clock) Page-Replacement Algorithm





Second-Chance (clock) Page-Replacement Algorithm



- We can see notably that the bad replacement decisions made by FIFO is not present in Second chance !!!
- There are a total of 9 page read operations to satisfy the total of 18 page requests - just as good as the more computationally expensive LRU method !!!

Enhanced Second-Chance Algorithm

- Improve algorithm by using reference bit and modify bit (if available) in concert

- Take ordered pair (reference, modify)

1. (0, 0) neither recently used not modified – best page to replace
2. (0, 1) not recently used but modified – not quite as good, must write out before replacement
3. (1, 0) recently used but clean – probably will be used again soon
4. (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

Counting Algorithms

- Keep a counter of the number of references that have been made to each page
  - Not common

- **Lease Frequently Used (LFU) Algorithm:** replaces page with smallest count

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

OPERATING SYSTEMS

Counting Algorithms - Least Frequently Used Algorithm Page Replacement Algorithm

Req #	R1	R2	R3	R4	R5	R6	R7	R9	R10
Page #	7	0	1	2	0	3	0	3	7

Fr #	R1	FQ
1	7	1
2		
3		
Flag	PF	NA

Fr #	R2	FQ
1	7	1
2	0	1
3		
Flag	PF	NA

Fr #	R3	FQ
1	7	1
2	0	1
3	1	1
Flag	PF	NA

Fr #	R4	FQ
1	2	1
2	0	1
3	1	1
Flag	PF	NA

Fr #	R5	FQ
1	2	1
2	0	2
3	1	1
Flag	PH	NA

Fr #	R6	FQ
1	2	1
2	0	2
3	3	1
Flag	PF	NA

Fr #	R7	FQ
1	2	1
2	0	3
3	3	2
Flag	PH	NA

Fr #	R10	FQ
1	7	1
2	0	3
3	3	2
Flag	PF	NA

Working Set => { }

Total Number of Page Requests = > 7

Total Number of Frames => 3

Total Number of Page faults =>

% of Page Faults = > => %

% of Page Hits => %

Legend

Page Fault => PF

Page Hit => PH

FQ => Frequency Count

OPERATING SYSTEMS

Counting Algorithms - Most Frequently Used Algorithm Page Replacement Algorithm

Req #	R1	R2	R3	R4	R5	R6	R7	R8
Page #	7	0	1	0	7	7	2	7

Fr #	R1	FQ
1	7	1
2		
3		
Flag	PF	NA

Fr #	R2	FQ
1	7	1
2	0	1
3		
Flag	PF	NA

Fr #	R3	FQ
1	7	1
2	0	1
3	1	1
Flag	PF	NA

Fr #	R4	FQ
1	7	1
2	0	2
3	1	1
Flag	PH	NA

Fr #	R5	FQ
1	7	2
2	0	2
3	1	1
Flag	PH	NA

Fr #	R6	FQ
1	7	3
2	0	2
3	1	1
Flag	PH	NA

Fr #	R7	FQ
1	2	1
2	0	2
3	1	1
Flag	PF	NA

Working Set => { }

Total Number of Page Requests = > 22

Total Number of Frames => 3

Total Number of Page faults =>

% of Page Faults = > => %

% of Page Hits => %

Legend

Page Fault => PF

Page Hit => PH

FQ => Frequency Count

Fr #	R7	FQ
1	2	1
2	7	1
3	1	1
Flag	PF	NA

Page-Buffering Algorithms

- Keep a pool of free frames, always
  - Then frame available when needed, not found at fault time
  - Read page into free frame
  - Select victim to evict and add to freed pool
  - When convenient, evict Victim

- Possibly, keep list of modified pages
  - When backing store otherwise idle, write pages there and set to non-dirty
  - Possibly, keep free frame contents intact and note what is in them
  - If referenced again before reused, no need to load contents again from disk
  - Generally useful to reduce penalty if wrong victim frame selected

Applications and Page Replacement

- All of these algorithms have OS guessing about future page access
- Some applications have better knowledge – i.e. databases
- Memory intensive applications can cause double buffering
  - OS keeps copy of page in memory as I/O buffer
  - Application keeps page in memory for its own work
- Operating system can given direct access to the disk, getting out of the way of the applications
  - Raw disk mode
- Bypasses buffering, locking, etc

Allocation of Frames

- Each process needs minimum number of frames
- 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
- Maximum of course is total frames in the system

- Two major allocation schemes
    - fixed allocation
    - priority allocation

- Many variations

Allocation of Frames

- Equal allocation – For example, if there are 100 frames (after allocating frames for the OS) and 5 processes, give each process 20 frames
  - Keep some as free frame buffer pool

- Proportional allocation – Allocate according to the size of process

- Dynamic as degree of multiprogramming, process sizes
  m = 64 change

$$s_i = \text{size of process } p_i$$
$$S = \sum s_i$$
$$m = \text{total number of frames}$$
$$a_i = \text{allocation for } p_i = \frac{s_i}{S} \times m$$

$$m = 64$$
$$s_1 = 10$$
$$s_2 = 127$$
$$a_1 = \frac{10}{137} \times 62 \approx 4$$
$$a_2 = \frac{127}{137} \times 62 \approx 57$$

Priority Allocation

- Use a proportional allocation scheme using priorities rather than size

- If process Pi 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
    - But then process execution time can vary greatly
    - But greater throughput so more common

Non-Uniform Memory Access

- So far all memory accessed equally
- Many systems are NUMA – speed of access to memory varies
  - Consider system boards containing CPUs and memory, interconnected over a system bus
- Optimal performance comes from allocating memory "close to" the CPU on which the thread is scheduled
- And modifying the scheduler to schedule the thread on the same system board when possible

Non-Uniform Memory Access

- Solved by Solaris by creating lgroups
  - Structure to track CPU / Memory low latency groups
  - Used my schedule and pager
  - When possible schedule all threads of a process and allocate all memory for that process within the lgroup

Topic Outline

- Second-Chance (Clock) Page-Replacement Algorithm
- Enhanced Second-Chance Algorithm
- Counting Algorithms
- Page-Buffering Algorithms

Topic Outline

- Applications and Page Replacement
- Allocation of Frames
- Fixed Allocation
- Priority Allocation
- Global vs. Local Allocation
- Non-Uniform Memory Access

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com and complete reading assignments provided by the Anchor on Edmodo