OPERATING SYSTEMS

Memory Management - 10

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

OPERATING SYSTEMS

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
- 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

Page Replacement

- Prevent over-allocation of memory by modifying page-fault service routine to include page replacement

- Use modify (**dirty**) bit to reduce overhead of page transfers => only modified pages are written to disk

- Page replacement completes separation between logical memory and physical memory => large virtual memory can be provided on a smaller physical memory

Page Replacement

1. Find the location of the desired page on disk

2. Find a free frame:

    i. If there is a free frame, use it

    ii. If there is no free frame, use a page replacement algorithm to

    iii. Select a **Victim** frame

        - Write **Victim** frame to disk if **dirty**

Page Replacement

3. Bring the desired page into the (newly) free frame; update the page and frame tables

4. Continue the process by restarting the instruction that caused the trap

Note: Potentially 2 page transfers for page fault – increasing EAT, if the page is swapped

Page and Frame Replacement Algorithms

- **Frame-allocation algorithm - FRA** determines
  - How many frames to give each process ?
  - Which frames to replace ?
- **Page-replacement algorithm - PRA**
  - Want lowest page-fault rate on both first access and re-access

Page and Frame Replacement Algorithms

- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string
  - String is just page numbers, not full addresses
  - Repeated access to the same page does not cause a page fault if available in the frame
  - Results depend on number of frames available
- In all our examples, the reference string of referenced page numbers of the same process is

Req #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21	R22
Page #	7	0	1	2	0	3	0	4	2	3	0	3	0	3	2	1	2	0	1	7	0	1

Optimal Page Replacement Algorithm

- Replace page that will not be used for longest period of time
- How do you know this ?
- Can’t read the future
- Used for measuring how well your algorithm performs

PRA : Optimal Page Replacement Algorithm

Req #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21	R22
Page #	7	0	1	2	0	3	0	4	2	3	0	3	0	3	2	1	2	0	1	7	0	1

Fr #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21	R22
1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
2		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0	0	0
3			1	1	1	3	3	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
Flag	PF	PF	PF	PF	PH	PF	PH	PF	PH	PH	PF	PH	PH	PH	PH	PF	PH	PH	PH	PF	PH	PH

Working Set => { 7,0,1,2,3,4 }

Total Number of Page Requests = > 22

Total Number of Frames => 3

Total Number of Page faults => 9

% of Page Faults = > 9/22 => 40.90%

% of Page Hits => 59.09 %

Legend
Page Fault => PF

Page Hit => PH

Least Recently Used Page Replacement Algorithm

- Use past knowledge rather than future
- Replace page that has not been used in the most amount of time
- Associate time of last use with each page
- Generally good algorithm and frequently used
- But how to implement ?

OPERATING SYSTEMS

PRA : Least Recently Used Page Replacement Algorithm - LRU

Req #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21	R22
Page #	7	0	1	2	0	3	0	4	2	3	0	3	0	3	2	1	2	0	1	7	0	1

Fr #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21	R22
1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	1	1	1	1	1	1	1
2		0	0	0	0	0	0	0	0	3	3	3	3	3	3	3	3	0	0	0	0	0
3			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	2	2	7	7	7
Flag	PF	PF	PF	PF	PH	PF	PH	PF	PF	PF	PF	PH	PH	PH	PH	PF	PH	PF	PH	PF	PH	PH

Working Set => { 7,0,1,2,3,4 }

Total Number of Page Requests = > 22

Total Number of Frames => 3

Total Number of Page faults => 12

% of Page Faults = > 12/22 => 54.54%

% of Page Hits => 45.45%

Legend

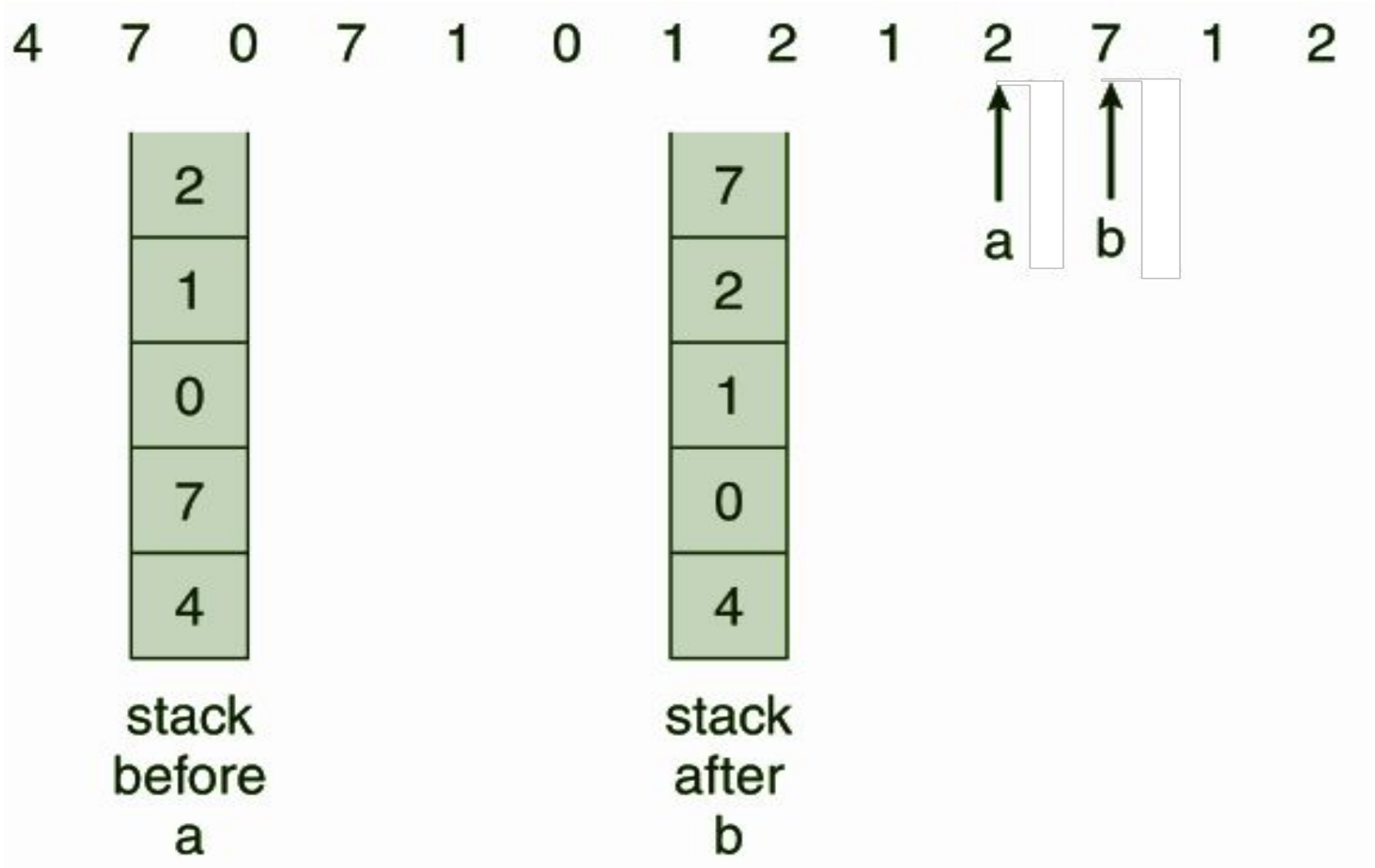
Page Fault => PF

Page Hit => PH

LRU Algorithm Implementation

- Counter implementation
    - Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter
    - When a page needs to be changed, look at the counters to find smallest value
        - Search through table needed
- Stack implementation
- Keep a stack of page numbers in a double link form:
- Page referenced: move it to the top
    - Requires 6 pointers to be changed
    - But each update more expensive
- No search for replacement
- LRU and OPT are cases of stack algorithms that don't suffer from Belady's Anomaly

Least Recently Used Page Replacement Algorithm



Page Replacement Algorithms - Comparison

First in First Out Page Replacement Algorithm

Working Set => { 7, 0, 1, 2 ,3, 4 }

Total Number of Page Requests = > 22

Total Number of Frames => 3

Total Number of Page faults => 15

% of Page Faults = > 15/22 => 68.18%

% of Page Hits => 31.81 %

Legend
Page Fault => PF

Page Hit => PH

Optimal Page Replacement Algorithm

Working Set => { 7,0,1,2,3,4 }

Total Number of Page Requests = > 22

Total Number of Frames => 3

Total Number of Page faults => 9

% of Page Faults = > 9/22 => 40.90%

% of Page Hits => 59.09 %

Legend
Page Fault => PF

Page Hit => PH

Least Recently Used Page Replacement Algorithm

Working Set => { 7,0,1,2,3,4 }

Total Number of Page Requests = > 22

Total Number of Frames => 3

Total Number of Page faults => 12

% of Page Faults = > 12/22 => 54.54%

% of Page Hits => 45.45%

Legend
Page Fault => PF

Page Hit => PH

Optimal Page Replacement Algorithm - Practice Problem

Req #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Page #	1	2	3	4	1	2	5	1	2	3	4	5

Fr #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
1												
2												
3												
Flag												

Working Set => { }

Total Number of Page Requests = >

Total Number of Frames =>

Total Number of Page faults =>

% of Page Faults = >

% of Page Hits =>

Legend
Page Fault => PF

Page Hit => PH

Optimal Page Replacement Algorithm - Practice Problem

Req #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Page #	1	2	3	4	1	2	5	1	2	3	4	5

Fr #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
1												
2												
3												
4												
Flag												

Working Set => { }

Total Number of Page Requests = >

Total Number of Frames =>

Total Number of Page faults =>

% of Page Faults = >

% of Page Hits =>

Legend

Page Fault => PF

Page Hit => PH

OPERATING SYSTEMS

Least Recently Used Page Replacement Algorithm - Practice Problem

Req #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Page #	1	2	3	4	1	2	5	1	2	3	4	5

Fr #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
1												
2												
3												
Flag												

Working Set => { }

Total Number of Page Requests = >

Total Number of Frames =>

Total Number of Page faults =>

% of Page Faults = >

% of Page Hits =>

Legend
Page Fault => PF

Page Hit => PH

OPERATING SYSTEMS

Least Recently Used Page Replacement Algorithm - Practice Problem

Req #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Page #	1	2	3	4	1	2	5	1	2	3	4	5

Fr #	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
1												
2												
3												
4												
Flag												

Working Set => { }

Total Number of Page Requests = >

Total Number of Frames =>

Total Number of Page faults =>

% of Page Faults = >

% of Page Hits =>

Legend

Page Fault => PF

Page Hit => PH

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
- LRU Algorithm Implementation

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