Chapter 11 – Virtual Memory Management

Chapter 11 – Virtual Memory Management

Objectives

- After reading this chapter, you should understand:
  - the benefits and drawbacks of demand and anticipatory paging.
  - the challenges of page replacement.
  - several popular page-replacement strategies and how they compare to optimal page replacement.
  - the impact of page size on virtual memory performance.
  - program behavior under paging.

11.1 Introduction

- Replacement strategy
  - Technique a system employs to select pages for replacement when memory is full
  - Determines where in main memory to place an incoming page or segment
- Fetch strategy
  - Determines when pages or segments should be loaded into main memory
  - Anticipatory fetch strategies
    - Use heuristics to predict which pages a process will soon reference and load those pages or segments

11.2 Locality

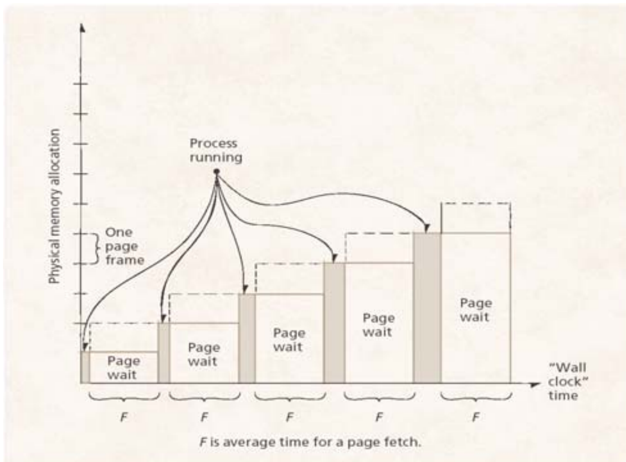
- Process tends to reference memory in highly localized patterns
  - In paging systems, processes tend to favor certain subsets of their pages, and these pages tend to be adjacent to one another in process’s virtual address space

11.3 Demand Paging

- Demand paging
  - When a process first executes, the system loads into main memory the page that contains its first instruction
  - After that, the system loads a page from secondary storage to main memory only when the process explicitly references that page
  - Requires a process to accumulate pages one at a time

11.3 Demand Paging

Figure 11.1 Space-time product under demand paging.



11.4 Anticipatory Paging

- Anticipatory paging
  - Operating system attempts to predict the pages a process will need and preloads these pages when memory space is available
  - Anticipatory paging strategies
    - Must be carefully designed so that overhead incurred by the strategy does not reduce system performance

11.5 Page Replacement

- When a process generates a page fault, the memory manager must locate referenced page in secondary storage, load it into page frame in main memory and update corresponding page table entry
- Modified (dirty) bit
  - Set to 1 if page has been modified; 0 otherwise
  - Help systems quickly determine which pages have been modified
- Optimal page replacement strategy (OPT or MIN)
  - Obtains optimal performance, replaces the page that will not be referenced again until furthest into the future

10.6 Page-Replacement Strategies

- A page-replacement strategy is characterized by
  - Heuristic it uses to select a page for replacement
  - The overhead it incurs

© 2004 Deitel & Associates, Inc. All rights reserved.

10.6.1 Random Page Replacement

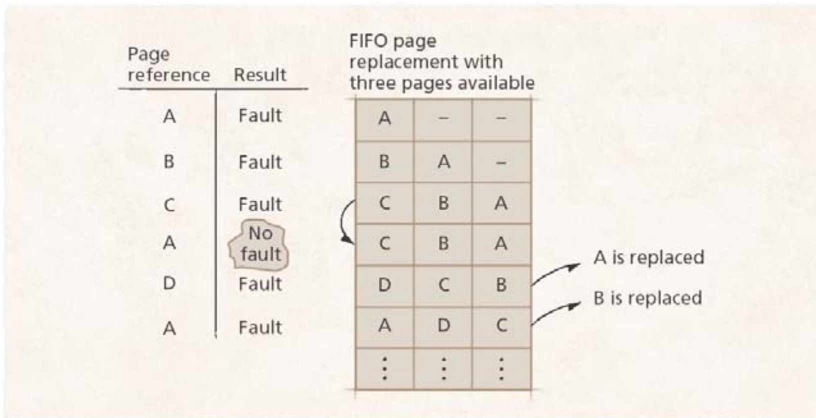
- Random page replacement
  - Low-overhead page-replacement strategy that does not discriminate against particular processes
  - Each page in main memory has an equal likelihood of being selected for replacement
  - Could easily select as the next page to replace the page that will be referenced next

10.6.2 First-In-First-Out (FIFO) Page Replacement

- FIFO page replacement
  - Replace page that has been in the system the longest
  - Likely to replace heavily used pages
  - Can be implemented with relatively low overhead
  - Impractical for most systems

11.6.2 First-In-First-Out (FIFO) Page Replacement

Figure 11.2 First-in-first-out (FIFO) page replacement.



11.6.3 FIFO Anomaly

- Belady’s (or FIFO) Anomoly
  - Certain page reference patterns actually cause more page faults when number of page frames allocated to a process is increased

11.6.3 FIFO Anomaly

Figure 11.3 FIFO anomaly–page faults can increase with page frame allocation.

Page reference	Result	FIFO page replacement with three pages available				FIFO page replacement with four pages available			
A	Fault	A	–	–	Fault	A	–	–	–
B	Fault	B	A	–	Fault	B	A	–	–
C	Fault	C	B	A	Fault	C	B	A	–
D	Fault	D	C	B	Fault	D	C	B	A
A	Fault	A	D	C	No fault	D	C	B	A
B	Fault	B	A	D	No fault	D	C	B	A
E	Fault	E	B	A	Fault	E	D	C	B
A	No fault	E	B	A	Fault	A	E	D	C
B	No fault	E	B	A	Fault	B	A	E	D
C	Fault	C	E	B	Fault	C	B	A	E
D	Fault	D	C	E	Fault	D	C	B	A
E	No fault	D	C	E	Fault	E	D	C	B

Three "no faults" Two "no faults"

11.6.4 Least-Recently-Used (LRU) Page Replacement

- LRU page replacement
  - Exploits temporal locality by replacing the page that has spent the longest time in memory without being referenced
  - Can provide better performance than FIFO
  - Increased system overhead
  - LRU can perform poorly if the least-recently used page is the next page to be referenced by a program that is iterating inside a loop that references several pages

11.6.4 Least-Recently-Used (LRU) Page Replacement

Figure 11.4 Least-recently-used (LRU) page-replacement strategy.

Page reference	Result	LRU page replacement with three pages available		
A	Fault	A	–	–
B	Fault	B	A	–
C	Fault	C	B	A
B	No fault	B	C	A
B	No fault	B	C	A
A	No fault	A	B	C
D	Fault	D	A	B
A	No fault	A	D	B
B	No fault	B	A	D
F	Fault	F	B	A
B	No fault	B	F	A

11.6.5 Least-Frequently-Used (LFU) Page Replacement

- LFU page replacement
  - Replaces page that is least intensively referenced
  - Based on the heuristic that a page not referenced often is not likely to be referenced in the future
  - Could easily select wrong page for replacement
    - A page that was referenced heavily in the past may never be referenced again, but will stay in memory while newer, active pages are replaced

11.6.6 Not-Used-Recently (NUR) Page Replacement

- NUR page replacement
  - Approximates LRU with little overhead by using referenced bit and modified bit to determine which page has not been used recently and can be replaced quickly
  - Can be implemented on machines that lack hardware referenced bit and/or modified bit

11.6.6 Not-Used-Recently (NUR) Page Replacement

Figure 11.5 Page types under NUR.

Group	Referenced	Modified	Description
Group 1	0	0	Best choice to replace
Group 2	0	1	[Seems unrealistic]
Group 3	1	0	
Group 4	1	1	Worst choice to replace

11.6.7 Modification to FIFO: Second-Chance and Clock Page Replacement

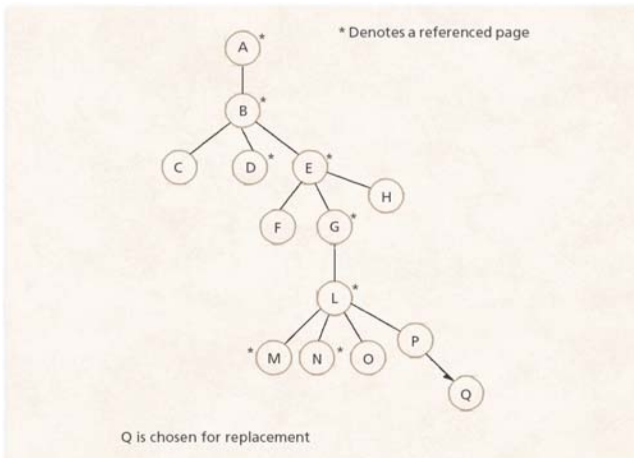
- Second chance page replacement
  - Examines referenced bit of the oldest page
    - If it's off
      - The strategy selects that page for replacement
    - If it's on
      - The strategy turns off the bit and moves the page to tail of FIFO queue
  - Ensures that active pages are the least likely to be replaced
- Clock page replacement
  - Similar to second chance, but arranges the pages in circular list instead of linear list

11.6.8 Far Page Replacement

- Far page replacment
  - Creates an access graph that characterizes a process's reference patterns
  - Replace the unreferenced page that is furthest away from any referenced page in the access graph
  - Performs at near-optimal levels
  - Has not been implemented in real systems
    - Access graph is complex to search and manage without hardware support

11.6.8 For Page Replacement

Figure 11.6 Far page-replacement-strategy access graph.

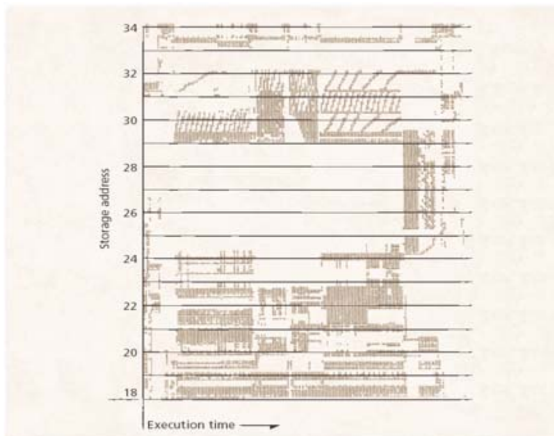


11.7 Working Set Model

- For a program to run efficiently
  - The system must maintain that program's favored subset of pages in main memory
- Otherwise
  - The system might experience excessive paging activity causing low processor utilization called thrashing as the program repeatedly requests pages from secondary storage

11.7 Working Set Model

Figure 11.7 Storage reference pattern exhibiting locality. (Reprinted by permission from IBM Systems Journal. © 1971 by International Business Machines Corporation.)

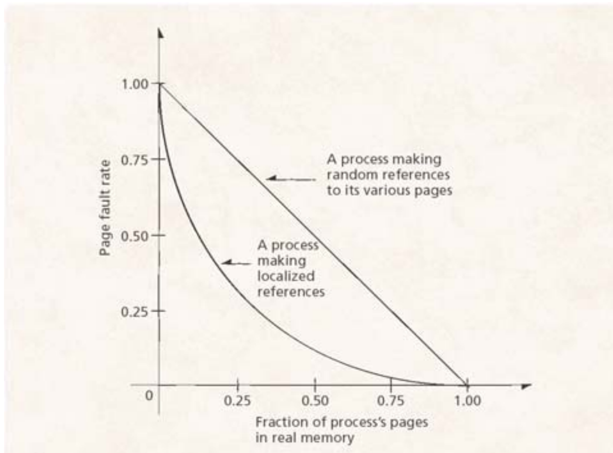


11.7 Working Set Model

- Because processes exhibit locality, increasing the number of page frames allocated to a process has little or no effect on its page fault rate at a certain threshold

11.7 Working Set Model

Figure 11.8 Dependence of page fault rate on amount of memory for a process's pages.

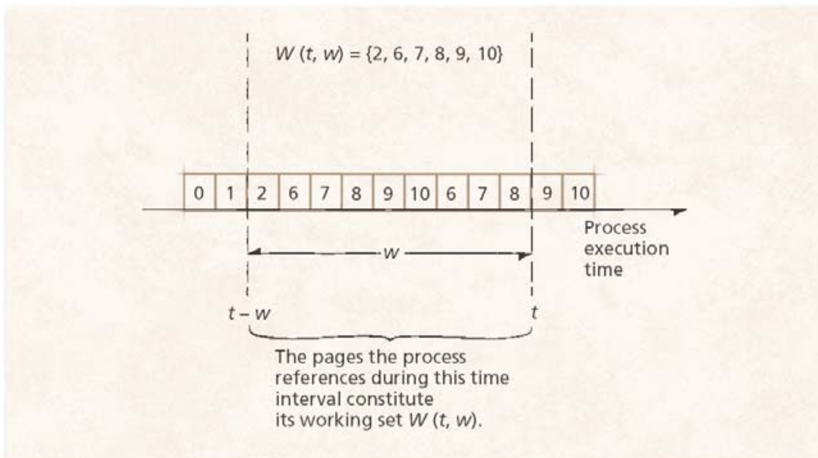


11.7 Working Set Model

- The process's working set of pages, $W(t, w)$, is the set of pages referenced by the process during the process-time interval $t – w$ to t.

11.7 Working Set Model

Figure 11.9 Definition of a process's working set of pages.

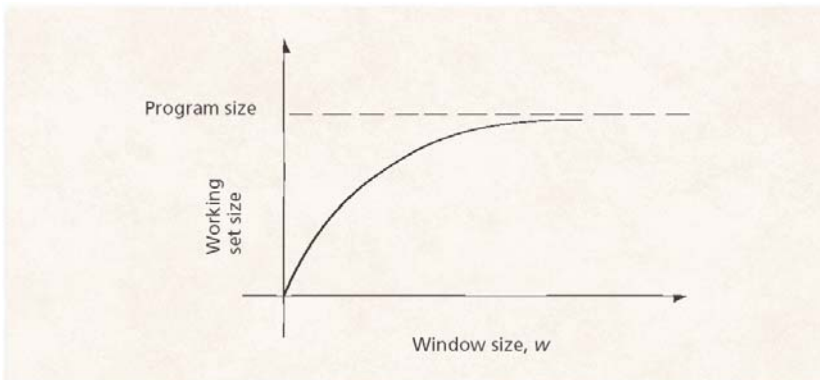


11.7 Working Set Model

- The size of the process's working set increases asymptotically to the process's program size as its working set window increases

11.7 Working Set Model

Figure 11.10 Working set size as a function of window size.

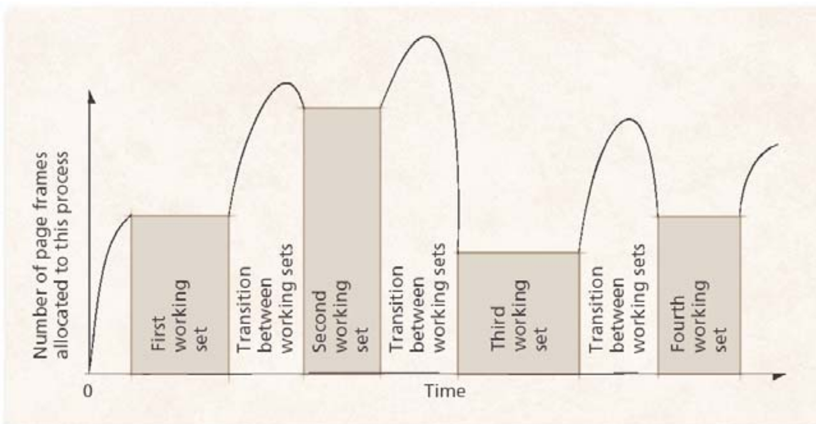


11.7 Working Set Model

- As a process transitions between working sets, the system temporarily maintains in memory pages that are no longer in the process's current working set
  - Goal of working set memory management is to reduce this misallocation

11.7 Working Set Model

Figure 11.11 Main memory allocation under working set memory management.



11.8 Page-Fault-Frequency (PFF) Page Replacement

- Adjusts a process's resident page set
  - Based on frequency at which the process is faulting
  - Based on time between page faults, called the process's interfault time
- Advantage of PFF over working set page replacement
  - Lower overhead
    - PFF adjusts resident page set only after each page fault
    - Working set mechanism must operate after each memory reference

11.9 Page Release

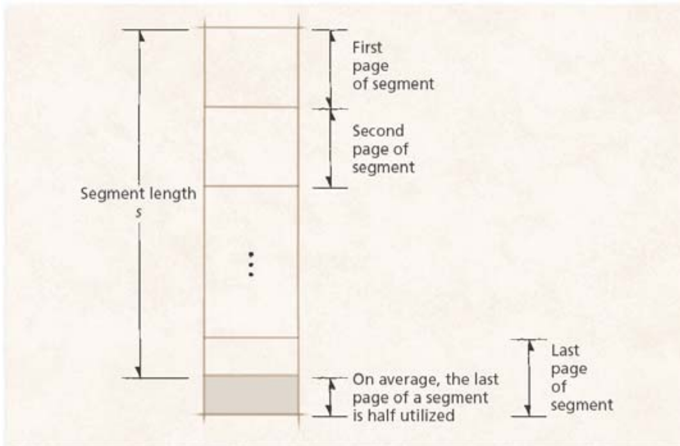
- Inactive pages can remain in main memory for a long time until the management strategy detects that the process no longer needs them
  - One way to solve the problem
    - Process issues a voluntary page release to free a page frame that it knows it no longer needs
    - Eliminate the delay period caused by letting process gradually pass the page from its working set
    - The real hope is in compiler and operating system support

11.10 Page Size

- Some systems improve performance and memory utilization by providing multiple page sizes
  - Small page sizes
    - Reduce internal fragmentation
    - Can reduce the amount of memory required to contain a process's working set
    - More memory available to other processes
  - Large page size
    - Reduce wasted memory from table fragmentation
    - Enable each TLB entry to map larger region of memory, improving performance
    - Reduce number of I/O operations the system performs to load a process's working set into memory
  - Multiple page size
    - Possibility of external fragmentation

11.10 Page Size

Figure 11.12 Internal fragmentation in a paged and segmented system.



11.10 Page Size

Figure 11.13 Page sizes in various processor architectures.

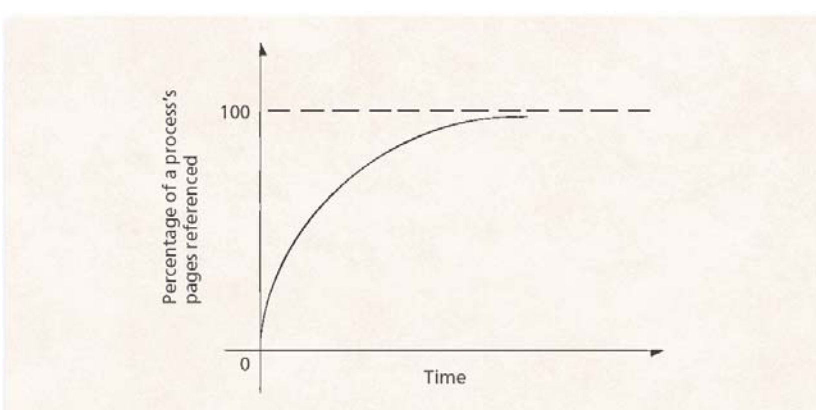
Manufacturer	Model	Page Size	Real address size
Honeywell	Multics	1KB	36 bits
IBM	370/168	4KB	32 bits
DEC	PDP-10 and PDP-20	512 bytes	36 bits
DEC	VAX 8800	512 bytes	32 bits
Intel	80386	4KB	32 bits
Intel / AMD	Pentium 4 / Athlon XP	4KB or 4MB	32- or 36 bits
Sun	UltraSparc II	8KB, 64KB, 512KB, 4MB	44 bits
AMD	Opteron / Athlon 64	4KB, 2MB and 4MB	32, 40, or 52 bits
Intel-HP	Itanium, Itanium 2	4KB, 8KB, 16KB, 64KB, 256KB, 1MB, 4MB, 16MB, 64MB, 256MB	Between 32 and 63 bits
IBM	PowerPC 970	4KB, 128KB, 256KB, 512KB, 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB, 256MB	32 or 64 bits

11.11 Program Behavior under Paging

- Processes tend to reference a significant portion of their pages within a short time after execution begins
- They access most of their remaining pages at a slower rate

11.11 Program Behavior under Paging

Figure 11.14 Percentage of a process's pages referenced with time.

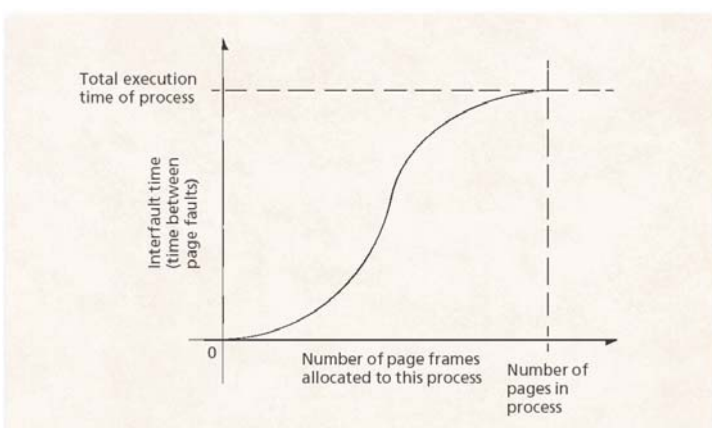


11.11 Program Behavior under Paging

- Average interfault time monotonically increases in general
  - The more page frames a process has, the longer the time between page faults

11.11 Program Behavior under Paging

Figure 11.15 Dependency of interfault time on the number of page frames allocated to a process.



11.12 Global vs. Local Page Replacement

- Implementing a paged virtual memory system
  - Global page-replacement strategies: applied to all processes as a unit
    - Tend to ignore characteristics of individual process behavior
    - Global LRU (gLRU) page-replacement strategy
      - Replaces the least-recently-used page in entire system
    - SEQ (sequence) global page-replacement strategy
      - Uses LRU strategy to replace pages until sequence of page faults to contiguous pages is detected, at which point it uses most-recently-used (MRU) page-replacement strategy
  - Local page-replacement strategies: Consider each process individually
    - Enables system to adjust memory allocation according to relative importance of each process to improve performance

11.13 Case Study: Linux Page Replacement

- Linux uses a variation of the clock algorithm to approximate an LRU page-replacement strategy
- The memory manager uses two linked lists
  - Active list
    - Contains active pages
    - Most-recently used pages are near head of active list
  - Inactive list
    - Contains inactive pages
    - Least-recently used pages near tail of inactive list
  - Only pages in the inactive list are replaced

11.13 Case Study: Linux Page Replacement

Figure 11.16 Linux page replacement overview.