

Segmentation

- Memory-management scheme that supports user view of memory.
- A program is a collection of segments. A segment is a logical unit such as:
 - main program,
 - Procedure,
 - function,
 - local variables, global variables,
 - common block,
 - stack,
 - symbol table, arrays

Segmentation

- Segments are variable-length modules of a program that correspond to logical units of the program, that is, segments represent the modular structure of how a program is organized.
- Each segment is actually a different logical address space of the program. Examples of segments are the program's main function, additional functions, data structures.

Segmentation and Memory Allocation

- Before a program can execute, all its segments need to be loaded into memory.
- For every segment, the operating system needs to find a contiguous block of available memory to allocate to the segment.

Virtual Memory

- The memory space of a process is normally divided into blocks that are either pages or segments.
- Virtual memory management takes advantage of the typical behavior of a process: not all blocks of the process are needed during the execution of a process.
- Therefore, the physical address space of a process is smaller than its logical address space.

Virtual Memory (2)

- Therefore, not all block of a process need main memory allocation.
- The physical address space of a process is smaller than its logical address space.
- For example, a process has 15 pages and only 7 frames allocated.

Virtual Memory Principles

A process can execute without having all its pages in physical memory. Some advantages are:

- A user process can be larger than physical memory
- Higher degree of multiprogramming
- Less I/O for loading and unloading for individual user processes
- Higher CPU utilization and throughput.

Address Space

- The virtual address space of a process is the entire set of all its addresses in the absolute program.
- After linkage, the absolute version of the program is stored on disk. The disk area that stores all the processes in absolute form is called the virtual memory.
- The physical address space of a process is much smaller than its virtual address because only a portion of the process will ever be loaded into main memory.

Referencing A Page

- A page reference is the page that has the address being referenced.
- The virtual memory manager swaps in a page of an executing process whenever the execution of a process references a page that is not in physical memory.
- Any unused page in physical memory will normally be swapped out to disk.

Virtual Memory Techniques

- Overlays (old technique)
- Paged virtual memory
- Segmented virtual memory

Paged Virtual Memory

When to swap pages into memory:

- Demand paging- a page is not swapped in until it is referenced
- Prepaging - a page is swapped in before it is referenced

Demand Paging

Load a page into memory only when it is needed (when it is referenced):

- Less I/O needed
- Less memory needed
- Faster response
- More users

Valid/Invalid Bit in Demand Paging

For every page referenced, a check of the resident bit is carried out

- invalid reference \Rightarrow abort
- not-in-memory \Rightarrow load page into physical memory

Valid-Invalid Bit

- With each page table entry a valid–invalid bit is associated:
 - 1 (in-memory)
 - 0 (not-in-memory)
- Initially valid–invalid but is set to 0 on all entries.

Frame #	valid-invalid bit
	1
	1
	1
	1
	0
⋮	
	0
	0

page table

Example of a page table snapshot.

Page Fault

Each page table entry has a resident bit, it indicates whether the corresponding page is currently in memory.

- If the page is not in memory, a page fault has occurred and the control is trapped to the OS.
- During address translation, if valid–invalid bit in page table entry is 0, a page fault has occurred.

Demand Paging

In demand paging, a page fault occurs when a reference is made to a page not in memory. The page fault may occur while:

- fetching an instruction, or
- fetching an operand of an instruction.

Performance in Virtual Memory

The performance of a virtual memory management system depends on the total number of page faults, which depend on:

- The paging policies
- The frame allocation

Types of Page Allocation

- Static allocation - the number of frames allocated to a process is fixed
- Dynamic allocation - the number of frames allocated to a process changes

Paging Policies

- Fetch policy -- decides when a page should be loaded into memory
- Replacement policy -- decides which page in memory should be replaced
- Placement policy -- decides where in memory should a page be loaded

Page Faults and Performance Issues

A page fault requires the operating system to carry out the page fault service. The total time it takes to service a page fault includes several time components:

- The time interval to service the page fault interrupt.
- The time interval to store back (swap out) the replaced page to the secondary storage device.
- The time interval to load (swap in) the referenced page from the secondary storage device (disk unit).
- Delay in queuing for the secondary storage device.
- Delay in scheduling the process with the referenced page.

Page Replacement

- When there is a page fault, the referenced page must be loaded
- If there is no available frame in memory one page is selected for replacement
- If the selected page has been modified, it must be copied back to disk (swapped out)

Performance and Page Replacement

- Page replacement – there are several algorithms
 - For improved performance – select a replacement algorithm that will result in minimum number of page faults.
- Same pages may be referenced several times.

Handling of a Page Fault

1. For every page reference, the page table entry for the page referenced is examined. If the access is invalid, the process is terminated.
2. If the referenced page is not in memory, a page fault occurs and the OS interrupts the process
3. The OS handles the page fault
 1. Carries out the page replacement.
 2. Swaps out the replaced page
 3. Swaps in the referenced page
 4. Reschedules the process that caused the page fault
4. The instruction which caused the page fault is restarted when the process resumes execution.

Performance Goal in VM

- Directly dependent on the number of page faults
- It is very important to keep the page fault rate low
- How can this be accomplished?

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

FRAME ALLOCATION

Allocation schemes:

- **Equal allocation**
- **Proportional allocation (based on size, priority,)**

Global vs. local replacement

- **Global: a replacement frame can be selected from the set of all frames**
- **Local: a replacement frame for a process can be selected only from its own set of allocated frames.**

Problems to be Solved in Demand Paging

Two major problems must be solved to implement demand paging:

- Each process needs a minimum number of frames. This minimum number is based on the machine architecture.
- Frame allocation - decide how many frames to allocate to each process.
- Select which frames are to be replaced when a page fault occurs.

Types of Paging Algorithms

- Static allocation - the number of frames allocated to a process is fixed
- Dynamic allocation - the number of frames allocated to a process changes

Paging Policies

- Fetch policy -- decides when a page should be loaded into memory
- Replacement policy -- decides which page in memory should be replaced
- Placement policy -- decides where in memory should a page be loaded

Selection of a Replacement Algorithm

How is a particular page replacement algorithm selected?

- In general, select the one with the lowest page fault rate
- An algorithm is evaluated by running it on a particular string of memory references, and computing the number of page faults.

Page Replacement Algorithms

- Goal: to minimize the page fault rate
- Input:
 - size of the process in pages
 - number of page frames allocated
 - a set of page reference strings
- Output: number of page faults

Page Replacement Algorithms

- A page reference string is a sequence of page numbers in order of reference. A subsequence of the same page number can be reduced to a single occurrence.
- A page replacement algorithm is said to satisfy the inclusion property or is called a stack algorithm if the set of pages in a k -frame memory is always a subset of the pages in a $(k + 1)$ -frame memory.

Stack Algorithms

- Every Process generates a sequence of memory references as it runs.
- Each memory reference corresponds to a specific virtual page.
- A process' memory access may be characterized by an ordered list of page numbers.
 - Ordered list referred to as the Reference String.
- To simplify: remainder of discussion centers around a machine w/1 process.
- Paging System may be characterized by 3 items:
 - The Reference String.
 - The Page Replacement Algorithm.
 - The number of page frames available in memory, m .

Page and Frame Tables

Contents of each entry in a page table:

- valid/invalid bit
- resident (or presence) bit
- frame number
- disk address

Contents of each entry in a frame table:

- mode: free, used, in transition, or locked in
- page-description address
- dirty bit: whether the page has been modified
- reference bit.

Page Reference

A page reference string is a sequence of page numbers in order of reference. A subsequence of the same page number can be reduced to a single occurrence.

- An example of a sequence of page references is:
< 3,6,2,1,4,7,3,5,8,9,2,8,10,7 >
- The sequence of page references represents the behavior of a process during execution.

Static Replacement Algorithms

- The static paging algorithms implement the replacement policy when the frame allocation to a process is fixed.
- The three most common static paging algorithms:
 - First-in-first-out (FIFO) replacement
 - Optimal replacement
 - Least recently used (LRU) replacement

FIFO Algorithm

- When a page fault occurs and there are no empty frames for the process, the page selected to be replaced is the one that has been in memory the longest time, the oldest page.
- This selection of the page to replace is completely independent of the locality of the process.
- Thus, it does not follow very well the general behavior of processes.

FIFO Example

Page reference stream:

1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3

1 1 1 1 1 2 2 3 5 1 6 6 2 5 5 3 3 1 6 2

2 2 2 2 3 3 5 1 6 2 2 5 3 3 1 1 6 2 4

3 3 3 5 5 1 6 2 5 5 3 1 1 6 6 2 4 3

* * * * * * * * * * * * * * *

FIFO

Total 14 page faults

Optimal Algorithm

- The optimal algorithm for page replacement requires knowledge of the entire page reference stream in advance.
- When a page fault occurs and there are no empty frames for the process, the algorithm looks ahead in the page reference stream to find out about future references to the pages currently in physical memory.
- The approach used is to replace the page in memory that will not be used for the longest period.

Optimal Example

Page reference stream:

1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3

1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 2 2 2

2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 4 4

3 3 3 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3

* * * * * * * * * *

Optimal

Total 9 page faults

LRU Algorithm

- The least recently used (LRU) replacement algorithm replaces the page that has not been used for the longest period.
- The assumption is that recent page references give a good estimation of page references in the near future.
- When a page fault occurs and there are no empty frames the algorithm selects the least recently referenced page for replacement.

Approximation of LRU

Most methods employ reference (or used) bits, one for each frame. when a page is brought into memory, its reference bit is set to 0. whenever a page is referenced, its reference bit is set to 1 (by hardware) . The reference bit may be reset to 1 periodically.

Method 1. recording value of reference bits.

- Keep a byte counter for each frame**
- At regular intervals (say 100 msec), a timer interrupt occurs . The o.s. shifts the reference bit for each frame into the highest order (leftmost) bit of the corresponding counter. also, the o.s. resets all reference bits.**
- A higher value of the counter indicates a more recently used page.**

LRU Example

Page reference stream:

1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3

1 1 1 1 3 2 1 5 2 1 6 2 5 6 6 1 3 6 1 2

2 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4

3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3

* * * * * * * * * *

LRU

Total 11 page faults

Approximation to LRU (cont.)

METHOD 2: SECOND CHANCE METHOD

- **THE FRAME TABLE IS VIEWED AS A CIRCULAR QUEUE WITH A POINTER.**
- **WHEN A PAGE FAULTS OCCUR, THE POINTER ADVANCES UNTIL IT FINDS A FRAME WHOSE REFERENCE BIT IS 0.**
- **AS IT ADVANCES, IT RESETS EACH REFERENCE BIT TO 0. NOTE THAT A FRAME WITH REFERENCE BIT 1 IS GIVEN A SECOND CHANCE.**

OTHER METHODS:

- **CLOCK ALGORITHM. SIMILAR TO SECOND CHANCE, BUT INCLUDES ALSO THE DIRTY BIT.**
- **POOL OF FREE FRAMES. USED IN VAX/VMS WITH FIFO.**
- **LEAST/MOST FREQUENTLY USED.**

Frames Needed by a Process

A process should allocate enough frames for its current locality.

- If more frames are allocated, there is little improvement
- If less frames are allocated, there is very high page fault rate

Frame Allocation Problem

If a process does not have “enough” frames allocated, the page-fault rate would be very high. This leads to:

- Low CPU utilization
- The OS attempts to increase the degree of multiprogramming
- Thrashing -- a process spends most or all of its time swapping pages.

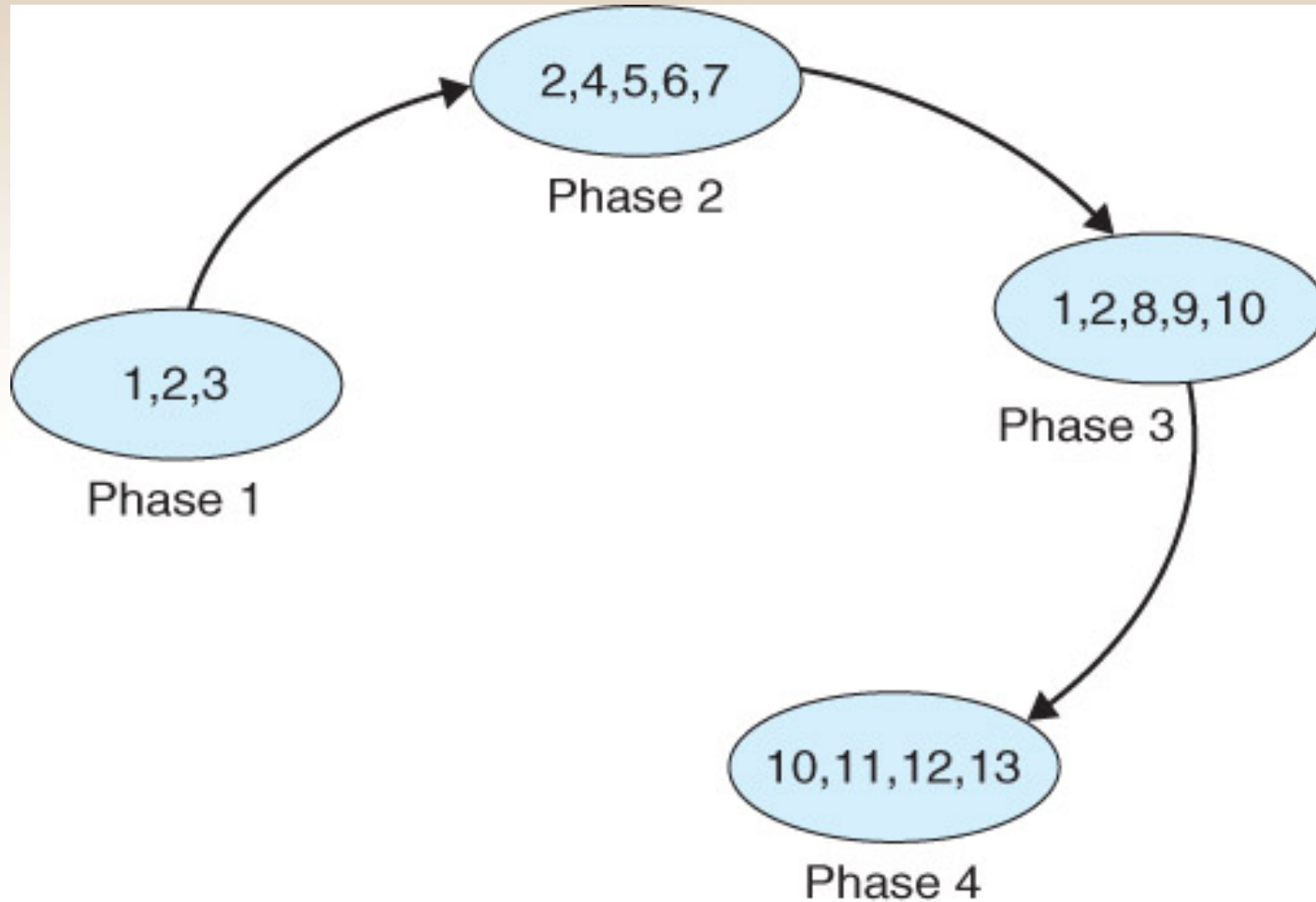
Thrashing

- A process is thrashing if it is spending more time paging than executing
- A thrashing process can cause other processes to thrash if a global page replacement strategy is used
- When CPU utilization is low, the OS may increase the degree of multiprogramming and cause other processes to thrash

Locality

- Why does paging work?
 - The locality model is used -- the set of pages used for that particular phase of computation
 - A process migrates from one locality to the next, as the process moves to a different phase of computation
 - Localities may overlap

Example of Locality



Locality of Program Execution

A set of pages that are actively used together.

- Temporal locality - once a location is referenced, it is often referenced again soon
- Spatial locality - once a location is referenced, a nearby location is often referenced soon

Locality and Thrashing

- Why does thrashing occur?
- The size of a locality is greater than the allocated memory (in frames)

Solve Thrashing

To stop thrashing, each active process should be allocated enough frames for its current locality

Dynamic Paging Algorithms

- The previous algorithms assume that a process is allocated a fixed amount of frames from the start of execution.
- Dynamic paging algorithms attempt to adjust the memory allocation to match the process' needs as it executes
- The Working Set algorithm is the best known algorithm for dynamic paging.

The Working Set Model

- The working set for each model is the set of pages referenced by the process during the most recent w page references
- The working set window is w , which is difficult to determine.
- Ideally, w is chosen so that at any time, the working set for a process is exactly the process' current locality

Total Demand of Frames

- Let d be the sum of the sizes of the working sets of all active processes.
- Let FA be the total number of frames
- If $d < FA$ then the OS can allow more active processes
- If $d > FA$ then the OS must suspend some active processes, otherwise thrashing occurs.

Page Fault Frequency Scheme

A strategy to control thrashing:

- Set the lower and upper bounds of page fault rate for each process
- Establish “acceptable” page fault rate
 - If actual rate is lower than lower bound, decrease the number of frames
 - If actual rate is larger than the upper bound, increase the number of frames