

Review

- Demand Paging
- Page Replacement
 - Optimal
 - FIFO
 - LRU
- Belady's Anomaly

NRU

- Page Classes (R, M)
 - (0,0) neither referenced nor dirty
 - (0,1) not referenced (recently) but dirtied
 - (1,0) referenced but clean
 - (1,1) referenced and dirtied
- Algorithm
 - select a page from lowest class
 - if conflict, use random or FIFO.

NFU: A LRU Approximation

- NFU (Not Frequently Used): Evict a page that is NOT frequently used;
LRU: evict a page that is LEAST recently used;
- NFU Implementation: simpler than LRU
 - additional reference bits
 - a counter is kept per page
 - a one bit is added into the counter if the page reference bit is set in a period
 - 00110011 would be accessed more frequently than 00010111
 - the page with the counter holding the lowest number is the least frequently used.
 - the value may not be unique. use FIFO to resolve conflicts.

Second Chance (Clock)

- Only one reference bit in the page table entry.
 - 0 initially
 - 1 When a page is referenced
- pages are kept in FIFO order
- Choose "victim" to evict
 - Select the head of FIFO
 - If page has reference bit set, reset bit, put it back to the tail of the list, and process the next page.
 - keep processing until reach page with zero reference bit and page that one out.
- Clock algorithm is a variant of second chance (use circular list to maintain the FIFO)

Frame Allocation: Minimum

- How are the page frames allocated to individual virtual memories in a multi-programmed environment?
- The simple case would be to allocate a minimal number of frames per process.
 - most instructions require two operands
 - include an extra page for paging out and one for paging in.
 - moves and indirection instructions might require more.

Equal Allocation

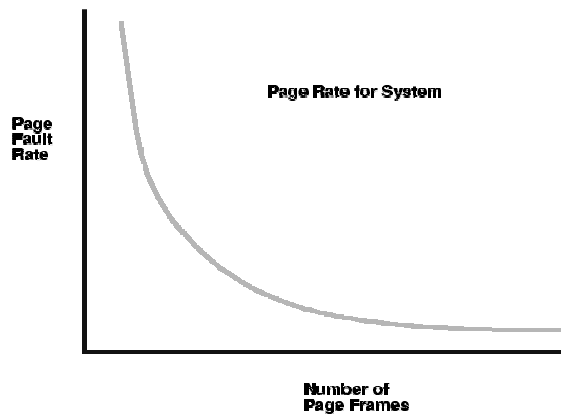
- allocate an equal number of frames per job
 - but jobs use memory unequally
 - high priority jobs have same number of page frames as low priority jobs
 - degree of multiprogramming might vary

Proportional Allocation

- allocate a number of frames per job proportional to job size
 - Challenge: how do you determine job size: by run command parameters or dynamically?

Page Fault Rate Curve

As page frames per VM space decrease, the page fault rate increases.

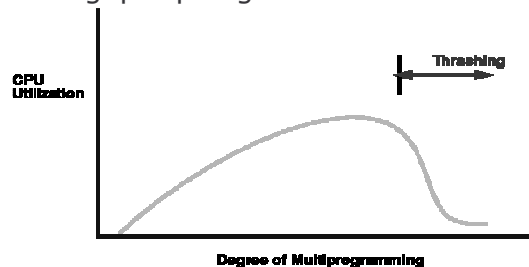


Thrashing

- Computation has locality.
- As page frames decrease, the page frames available are not large enough to contain the locality of the process.
- The processes start faulting heavily.
- Pages that are read in, are used and immediately paged out.

Thrashing and CPU Utilization

- As the page fault rate goes up, processes get suspended on page out queues for the disk.
- the system may try to optimize performance by starting new jobs.
- starting new jobs will reduce the number of page frames available to each process, increasing the page fault requests.
- system throughput plunges.

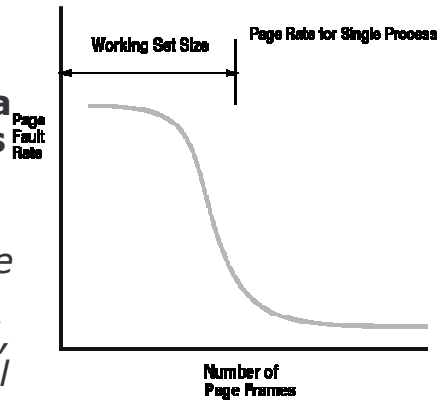


Working Set

□ the working set model assumes locality.

□ **the principle of locality states that a program clusters its access to data and text temporally.**

□ *As the number of page frames increases above some threshold, the page fault rate will drop dramatically.*



Working Set in Action

□ Algorithm

- if # free page frames > working set of some suspended process, then activate process and map in all its working set
- if working set size of some process increases and no page frame is free, suspend process and release all its pages

□ moving window over reference string used for determination.

□ keeping track of working set.

I/O Overview

I/O Software

- Interrupt Handlers, Device Driver, Device-Independent Software, User-Space I/O Software

Basic I/O hardware

- ports, buses, devices and controllers

Real I/O devices

- Disks, Character-Oriented Terminals, Graphical User Interfaces, Network Terminals

Device-Computer/Device-Device Communication

Physically

- via signals over a cable or through air

Logically

- via a connection point - port (e.g., serial port)

Multiple devices are connected via a bus

- A common set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires

Device Controller (I)

- I/O units typically consist of
 - a mechanical component, the device itself
 - an electronic component called the device controller or adapter.
- Interface between controller and device is a very low level interface.
- Example:
 - Disk's controller converts the serial bit stream, coming off the drive, into a block of bytes, and performs error correction. The block of bytes is first assembled in a buffer inside the controller. After its checksum has been verified, the error-free block is copied to main memory.
 - Built-in controllers

Device Controller (II)

- Example: Disk controller
 - implements the disk side of the protocol that does: bad error mapping, prefetching, buffering, caching
- Controller has registers for data and control
- CPU and controllers communicate via I/O instructions and registers

How to Perform I/O Operations?

- Programmed I/O
- Interrupt-driven I/O
- I/O using DMA

Programmed I/O

- Every I/O operation is programmed and controlled.
- Example:
 - printing a file from user program to the printer means that data is first copied to the kernel, then the OS enters a tight loop outputting the characters one at a time.
- Essential aspect
 - the CPU continuously polls the device to see if it is ready to accept another character. It uses **polling** (or busy waiting).
- Cons & Pros?
 - Pros: simple
 - Cons: required CPU full time until all the I/O is done.

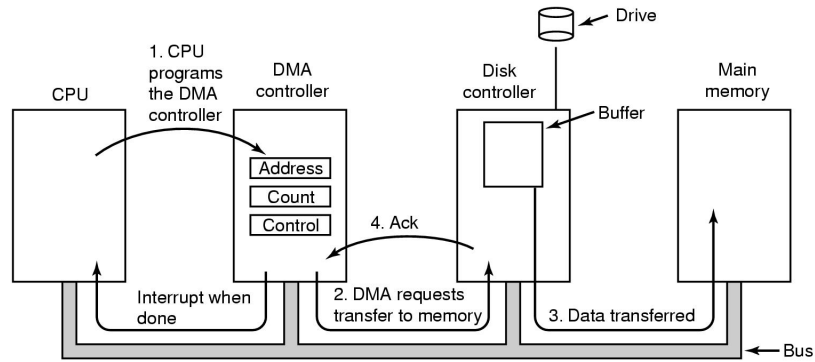
Interrupt-Driven I/O

- CPU hardware has the interrupt report line that the CPU senses after executing every instruction.
 - Device raises an interrupt
 - CPU catches the interrupt and saves the state (e.g., instruction pointer)
 - CPU dispatches the interrupt handler
 - Interrupt handler determines the cause of the interrupt, services the device and clears the interrupt.

Direct Memory Access (DMA)

- Use a special purpose processor, called a DMA controller
- Host writes a DMA command block into memory
 - pointer to source, pointer to destination, count of bytes to be transferred
- CPU writes the address of this command block to the DMA controller and goes on with other work
- DMA controller proceeds to operate the memory bus directly without help of main CPU
- Cons & Pros?

Direct Memory Access (DMA)



DMA controller feeds the characters to the printer one at the time, without CPU being bothered. DMA is actually the programmed IO, only with DMA controller doing the work.