Page Faults

- Sources to service page faults
  - faults through swap partitions
  - faults through executable files
  - faults through free page list
  - faults through modified page list
  - demand-zero faults
  - faults through mapped files

Other Facilities

- Demand-zero pages
- Copy-on-write:
  - places where it is used: forking a child process, transferring a large message
  - implementation:
    - special “copy-on-write” bit at the page table entries
    - mark the page as “read-only”
    - when write-permission fault, allocate a new page, and copy the content to the new page
    - change the page table entries

Paging in Kernel Space

- Kernel has pagable and nonpagable parts
  - pagable part has a system resident set and limit
  - normally kernel paged code and data are added onto system resident set
  - process page tables are added onto process’s resident size list
  - process page table not replacable if it contains valid page entries (why?)

VM in UNIX on VAX

- Why paging is necessary
- VAX hardware doesn’t have reference bits
- basic technique: simulate reference bit via valid bit and software fault handling

Problems with VMS’s Solution

- why they don’t like VMS’s solution
  - reliance on global free list introduces coupling effects
  - processes vary greatly in size and access pattern
  - processes’ page reference behavior changes, thus resident set limit should change
  - UNIX uses many processes to finish a task
- solution: the clock algorithm in Multics (“scan”)

Global Clock Replacement

- Reasons
  - simplicity and ease of implementation
  - no requirement of predictable performance in UNIX
  - users can’t specify what they need
  - no reference bits, can’t implement SWS, etc.
  - multiple processes for one task
- Problems

Free Page Pool and Clock Scan

- why free page pool is a good idea
- clock triggered whenever pool size < limit
- clock scan rate
- writebacks
- free pool size as load control

Controlling System Load

- when to initiate a swapout
  - can't maintain minimum free page pool while at maximum scan rate
- swap which process out
  - oldest of n largest resident processes
- when to let them return
  - based on size and time of being swapped out
  - don't swap in the whole resident set

Performance Comparison

- Purpose:
  - compare performance of the system with/without virtual memory
- Machine used:
  - VAX-11/780, 1MB main memory, two disks (1MB/s, 40ms avg access time)
- Applications:
  - script of four concurrent terminal sessions, staggered initiation of the scripts

Performance Results

- Data collected
  - average completion time of each terminal session
  - total completion time
  - average completion time
  - system CPU overhead
  - page traffic
- Reading the results (note page traffic)

Performance Enhancement

- Optimizations from the first experiments
  - Problems of small page size
  - forced to use clustering on page in and page out
- Optimization from comparing with VMS
  - clustering on page in and page out
  - where to put the pre-paged data
  - the "random" call