CIS 415: Operating Systems

VM Uses

Spring 2011
Prof. Butler
Last class:
  - Virtual Memory

Today:
  - Virtual Memory Uses
Efficient Physical Memory

- Through virtual memory…
  - $N \times 2^{32}$-sized address spaces
  - All isolated by default
- Uses for memory
  - Make a new process
    - Address space
  - Make an IPC
    - Or a cross-address space call
- Challenges in memory use
Shared Pages

- Shared code
  - One copy of read-only (*reentrant*) code shared among processes (i.e., text editors, compilers, window systems).

- Private code and data
  - Each process keeps a separate copy of the code and data
  - The pages for the private code and data can appear anywhere in the logical address space
Shared Pages Example
Create New Address Space

- **Via fork or clone**
  - Copy of the old address space
- **Change completely**
  - Exec
- **Or use the copy independently**
Copy-on-Write

- **Copy-on-Write** (COW) allows both parent and child processes to initially *share* the same pages in memory
  - If either process modifies a shared page, only then is the page copied
- COW allows more efficient process creation as only modified pages are copied
- Free pages are allocated from a **pool** of zeroed-out pages
Before Process 1 modifies Page C...
After Process 1 modifies Page C...
Memory-Mapped Files

- Memory-mapped file I/O allows file I/O to be treated as *routine memory access* by *mapping* a disk block to a page in memory
  - File is initially read using demand paging
  - Page-sized portion of the file is read from the file system into a physical page
  - Subsequent reads/writes to/from the file are treated as ordinary memory accesses.
Memory-Mapping Benefits

- Simplifies file access by treating file I/O through memory rather than `read()` or `write()` system calls
  - What is the benefit of doing this?
- Also allows several processes to map the same file allowing the pages in memory to be shared
Memory Mapped Files

[Diagram showing the concept of memory-mapped files, with processes A and B mapping disk file sections into their virtual memory spaces.]
Memory-Mapped Shared Mem

- process$_1$
  - shared memory
- memory-mapped file
- shared memory
- process$_2$
  - shared memory
Thrashing

- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
  - low CPU utilization
  - operating system thinks that it needs to increase the degree of multiprogramming
  - another process added to the system

- **Thrashing** ≡ a process is busy swapping pages in and out
Thrashing

![Graph showing CPU utilization vs. degree of multiprogramming with a peak and subsequent decline indicative of thrashing.](image)
Demand Paging & Thrashing

- Why does demand paging work?
  Locality model
  - Process migrates from one locality to another
  - Localities may overlap

- Why does thrashing occur?
  $\Sigma$ size of locality $>$ total memory size
Memory-Reference Locality
Working-Set Model

- $\Delta \equiv \text{working-set window} \equiv$ a fixed number of page references (e.g., 10,000 instructions)
- $WSS_i$ (working set of Process $P_i$) = total number of pages referenced in the most recent $\Delta$ (varies in time)
  - if $\Delta$ too small, will not encompass entire locality
  - if $\Delta$ too large, will encompass several localities
  - if $\Delta = \infty \Rightarrow$ will encompass entire program

- $D = \Sigma WSS_i \equiv$ total demand frames
- if $D > m \Rightarrow$ Thrashing
- Policy: if $D > m$, suspend one of the processes
Working-set model

Sliding window that approximates program locality
Tracking the Working Set

- Approximate with interval timer + reference bits
- Example: $\Delta = 10,000$
  - Timer interrupts after every 5000 time units
  - Keep in memory 2 bits for each page
  - Whenever a timer interrupts copy and set the values of all reference bits to 0
    - If one of the bits in memory $= 1 \Rightarrow$ page in working set
- Why is this not completely accurate?
- Improvement $= 10$ bits and interrupt every 1000 time units
Page-Fault Frequency

- Establish “acceptable” page-fault rate
  - If actual rate too low, process loses frame
  - If actual rate too high, process gains frame
Summary

- Uses
  - Shared Pages
  - Copy-on-write
  - Memory-mapped files

- Thrashing and the Working Set model
• Next time: Files