Virtual Memory (II)
CSCI 444/544 Operating Systems
Fall 2008

Agenda
- Memory allocation across multiple processes
- Thrashing
- Working set
- Trends

Memory Allocation Across Processes
- Physical pages allocated to several processes A, B, and C.
  - Process B page faults.
- Which page should be replaced?
- Three options
  - Per-process replacement
  - Per-user replacement
  - Global replacement

Per-process Replacement
- Each process has separate pool of pages
  - Fixed number of pages (e.g., Digital VMS)
  - Fixed fraction of physical memory (1/P)
  - Proportional to size of allocated address space
- Page fault in one process only replaces pages of that process
  - Perform replacement (e.g., LRU) over only those pages
- Advantage: No interference across processes
- Disadvantage: Potentially inefficient allocation of memory
Per-user Replacement

- Each user has separate pool of pages

- Advantage: Fair across different users

- Disadvantage: Inefficient allocation as well

Global Replacement

- Pages from all processes lumped into single replacement pool
  - Example: Run clock over all page frames

- Each process competes with other processes for frames

- Advantages:
  - Flexibility of allocation
  - Minimize total number of page faults

- Disadvantages
  - One memory-intensive process can hog memory, hurt all processes

Impact of Additional Processes

What happens to “performance” as add more processes?

- Consider CPU utilization as metric

- Increase number of processes from 1
  - Process blocks: Other processes can run
  - CPU utilization increases with more processes

- Increase number of processes after memory filled
  - Increases number of page faults
  - Memory contention increases with more processes
  - CPU utilization decreases with more processes

Thrashing

when the system spends most of its time servicing page faults, little time doing useful work

- Implication: Average memory access time = disk access time

- Memory appears as slow as disk, instead of disk appearing as fast as memory

- could be that there is enough memory but a lousy replacement algorithm (one incompatible with program behavior)

- could be that memory is over-committed
  - too many active processes
Thrashing (II)

Solution to Thrashing

Thrashing cannot be fixed with better replacement policies only

- Page replacement policies do not indicate that a page must be kept in memory
- Only show which pages are better than others to replace

Student’s analogy to thrashing: Too many courses

- Solution: Drop a course

OS solution: Admission control

- Determine how much memory each process needs
- Long-term scheduling policy

Working Set

Informal definition

- Collection of pages the process is referencing frequently
- Collection of pages that must be resident to avoid thrashing

Implication

- Locality exists
- Working set changes slowly over time

Example:

- Page reference stream:
  - A B A B C B A C A D C D E B E D F B F D B E D

Working Set Model of Program Behavior

The working set of a process is used to model the dynamic locality of its memory usage

- working set = set of pages process currently “needs”
- formally defined by Peter Denning in the 1960’s

Formal Definition:

- \( WS(t, w) = \{ \text{pages } P \text{ such that } P \text{ was referenced in the time interval } (t, t-w) \} \)
  - \( t \): time
  - \( w \): working set window (measured in page refs)
  - a page is in the working set (WS) only if it was referenced in the last \( w \) references

- obviously the working set (the particular pages) varies over the life of the program
- so does the working set size (the number of pages in the WS)
## Working Set Size

The working set size, \(|WS(t,w)|\), changes with program locality:
- During periods of poor locality, more pages are referenced.
- Within that period of time, the working set size is larger.

Intuitively, the working set must be in memory, otherwise you'll experience heavy faulting (thrashing):
- When people ask "How much memory does Firefox need?", really they're asking "what is Firefox's average (or worst case) working set size?"

## Balance Set

Motivation: Process should not be scheduled unless working set is resident in main memory.

Divide runnable processes into two groups:
- Active: Working set is loaded
- Inactive: Working set is swapped to disk

Balance set: Sum of working sets of all active processes.

Interaction with scheduler:
- If balance set exceeds size of memory, move some process to inactive set.
- If balance set is less than size of memory, move some process to active set.

## Trends

VM code is not as critical as before:
- Reason #1: Personal vs. time-shared machine
- Reason #2: Memory is more affordable, more memory

Less hardware support for replacement policies:
- Software emulation of reference and dirty bits

Larger page sizes:
- Better TLB coverage
- Smaller page tables
- Disadvantage: More internal fragmentation
  - Multiple page sizes