Virtual Memory

Virtual memory – separation of user logical memory from physical Memory

Virtual memory can be implemented via:

• Demand paging

• Demand segmentation

Demand Paging

Could bring entire process into memory at load time

• Or bring a page into memory only when it is needed

• Less I/O needed, no unnecessary I/O

• Less memory needed

• Faster response

• More users

• Lazy swapper – never swaps a page into memory unless page will be needed

• Swapper that deals with pages is a pager

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

• In general, free pages are allocated from a pool of zero-fill-on-demand pages

• vfork() variation on fork() system call has parent suspend and child using copy-on-write address space of parent

Page replacement

• Prevent over-allocation of memory by modifying page-fault service routine to include page replacement

• Use modify (dirty) bit to reduce overhead of page transfers – only modified pages are written to disk

• Page replacement completes separation between logical memory and physical memory – large virtual memory can be provided on a smaller physical memory

• Frame-allocation algorithm determines

• How many frames to give each process

• Which frames to replace

• Page-replacement algorithm

• Want lowest page-fault rate on both first access and re-acces

First-In-First-Out (FIFO) Algorithm

Optimal Algorithm

• Replace page that will not be used for longest period of time

Least Recently Used (LRU) Algorithm

• Use past knowledge rather than future

• Replace page that has not been used in the most amount of time

• Associate time of last use with each page

Counting-Based Page Replacement

• We can keep a counter of the number of references that have been made to each page

• Lease Frequently Used (LFU) Algorithm: replaces page with smallest count

• Most Frequently Used (MFU) Algorithm: based on the argument that the page with the smallest count was probably just brought in and has yet to be used

• Global replacement – process selects a replacement frame from the set of all frames; one process can take a frame from another

• But then process execution time can vary greatly

• But greater throughput so more common

• Local replacement – each process selects from only its own set of allocated frames

• More consistent per-process performance

• But possibly underutilized memory

Non-Uniform Memory Access

Many systems are NUMA – speed of access to memory varies

• Consider system boards containing CPUs and memory, interconnected over a system bus

• Optimal performance comes from allocating memory “close to” the CPU on which the thread is scheduled

• And modifying the scheduler to schedule the thread on the same system board when possible

• Thrashing a process is busy swapping pages in and out