Memory Management of Linux, Windows, and FreeBSD Operating Systems

Soo-Min Yoo - CS444 Spring 2017

Memory management is an essential, core part of an operating system. Operating systems use memory management to handle or manage primary memory and moves processes back and forth between the main memory and disk while executing. It keeps track of every memory location, whether it's either allocated or free. Memory management is implemented in different ways among different operating systems, and so in this paper we will be discussing the similarities and differences between how Linux, Windows, and FreeBSD implements it.

A important concept of memory management is virtual memory. Over many years, researchers noticed the growing need for more memory space for application programs, and the most successful solution was virtual memory. This makes the system appear to have more memory than it actually has, by using the hard disk to meet the extra space that's needed. Virtual memory is essentially a layer of memory addresses that map to physical addresses [1]. When a processor executes a program instruction, it reads from the virtual memory, converts the virtual memory address to a physical address, then executes it. This mapping of virtual to physical address is done based on mapping information in page tables maintained by the operating system [2].

Both virtual and physical address spaces are divided into fixed-size chunks called pages. The processor's Memory Management Unit (MMU) uses a page table to translate virtual memory address to a physical address, using a page table which specifies mapping between virtual pages and physical pages [2]. Virtual memory addresses are made up of an offset and a virtual page frame number. A processor takes this page frame number from a virtual page and translates it into a physical page frame number, and then uses the offset to put it in the correct address in the physical page [3]. There are cases where a processor tries to translate a virtual page and finds that the entry is invalid, or that since the virtual address space is much bigger than the physical space, pages are unable to be stored in the physical memory. These unstored pages are handled by the hard disk as page faults [3]. The page table associates each virtual page with a bit indicating whether or not it is present in physical memory. If not, the hard disk generates a page fault exception which returns an error if invalid or brings the required page from the hard disk into the physical memory. This time-consuming operation is called demand paging, where all memory pages in a process are not in the physical memory at any given time [1]. It's an efficient way to bring only the memory pages that are necessary at that moment into the physical memory, without the non-needed pages taking up space.

Linux, Windows, and FreeBSD operating systems have several similarities when it comes to memory management. All three systems have what's called a hardware abstraction layer, which takes care of all system-dependent work so that the remaining kernel parts are platform-independent, making it easy to port to other platforms [3]. When processes are sharing one page, a copy-on-write action occurs where a process doing a write on the page creates a private copy for that process. This action results in shadow paging, where a shadow object for the original object is created such that it alters some

pages from the original but shares the rest. The three systems also each have a background daemon that runs from time to time doing tasks like page flushing and freeing unused memory. Memory mapped files can be shared between processes as inter-process communication [3]. The three systems also distribute the process virtual address space quite similarly. The higher part is used by the kernel while the lower part is used by the process. Since the kernel part of the process space use the same kernel code, switching a process involves switching the lower part's page table entries while the higher part can stay the same [3].

Data structures are handled somewhat differently among the three systems. Linux maintains a linked list of `vm_area_structs`, which are continuous memory areas. This list is searched whenever a page with a specific location needs to be found. The list is turned into a tree if the number of entries grows big enough, and this allows for the most efficient structure to be used in the appropriate situations [3]. Windows uses a balanced tree with nodes called Virtual Address Descriptors (VAD), which marks each node as committed, free, or reserved. Committed nodes are used ones with code or data mapped to it, free nodes are unused ones, and reserved nodes cannot have anything mapped to it until its reserved mark is removed. Since it's the tree is balanced, finding a node with a specific location will also take relatively low search time. The Process Control Block contains the link to the tree's root [3]. FreeBSD's data structures consist of `vmspace`, `vm_map`, `vm_map_entry`, `object`, `shadow object`, and `vm_page`. The `vm_pmap` takes care of hardware-dependent work, which leaves the other parts of the VM hardware-independent and makes it efficient to port it to different platforms [3].

Page replacement is another important part of memory management, which involves choosing which page to swap out from memory when more free space is needed. Linux uses a demand paged system with no prepaging, where pages are only brought into memory when they're required [1]. They use the Least Recently Used algorithm to increase a page's age (a counter associated with the page) by a constant when it's used during a scan, and decrease it when it's not used. The pages with an age of 0 are removed from memory [3]. Windows uses clustered demand paging to fetch pages, and the clock algorithm to replace them. Pages are only brought into memory when they're needed, usually in a cluster of 1-8 pages. FreeBSD uses demand paging system with some prepaging for fetching and global Least Recently Used algorithm for replacing [3].

All three operating systems are similar in many ways regarding how they manage memory, while having a few differences here and there. I think the similarities and differences between these operating systems exist because Windows was developed to have better performance due to having had more effort and thought put into its design, whereas Unix-based systems like Linux and FreeBSD were designed more for the purpose of simplicity than performance. This resulted in Windows having more complex, intricate code with many features but more difficult to maintain; while Linux and FreeBSD are simpler and easier to maintain in comparison.

References

[1] A. Himanshu. (2012) Linux memory management – virtual memory and demand paging. [Online]. Available: http://www.thegeekstuff.com/2012/02/linux-memory-management

[2] R. Dube, "A comparison of the memory management sub-systems in freebsd and linux," Master's thesis, Univ. Maryland, College Park, 1998.

[3] G. Khetan, "Comparison of memory management systems of bsd, windows, and linux," Master's thesis, Univ. Southern California, Los Angeles, 2002.