Virtual memory and its architecture in operating systems 2287

INTRODUCTION

Virtual memory refers to an imaginary set of locations, or addresses, where you can store data. It is imaginary in the sense that the memory area is not the same as the real physical memory composed of transistors or capacitors. The efficient use of memory is a problem that has remained important since the development of the stored_program computer. At a time when computer memory was still expensive and had a small capacitance the programmers were encouraged to develop programs, that used less than expected memory. This problem led to one of the most ingenious ideas. In 1961 a group of people at Manchester, England proposed a method for performing the overlay process automatically which, in turn, led to virtual memory.

The concept of virtual memory is to greatly increase the apparent amount of physical memory available to programs. The free space on the computer's hard drive is treated as an extension of RAM (random access memory). When necessary, pages are swapped out of actual memory to a location on the hard drive. The swapped files can grow and shrink as thew memory demand fluctuates.

This concept is the key, which allows a user to run an operating system on a machine with 4, or 8 MB of installed RAM. On such systems, the operating system must swap portions of its self from the physical RAM to the hard drive's swap files. This can also be done as it loads. In addition, the virtual memory also utilizes the processor's ability to stop executing instructions when a page translation fails. In this paper the Unix, Apple, and the NT operating systems will be discussed.

There are three basic elements of virtual memory. Virtual address space, the main memory, and the auxiliary memory (Carr, 1984).

The address space is the set space, which is reserved for the use of memory swap. It creates a distinct location for data to be held. This process tremendously speeds up the retrieval time of virtual data. To the task, the address space is simply a continuous array of words or bytes, each with unique virtual address. To the virtual memory hardware, the address space is partitioned equal size frame or blocks (Carr, 1984). The auxiliary memory has a much longer access time but a lot less expensive than main memory. Conceptually, if one thinks of the program in the secondary memory as the original copy and the pieces brought in to the main memory as copies, rather the other way around (Tanenbaum, 1976).

Keeping the original program updated must not be over looked. When a program needs a page out of virtual memory, the operating system must find where that page is located. This task is simplified by partitioning the drive to allow virtual memory to be stored in a specific part of the hard drive.

The pages that are placed in frames inside virtual memory are known as the task's resident set. The resident set pages and the frames in which they reside is identified by a data structure called the Page table (Carr, 1984). The User can remove one or many pages out of the resident set. As a program executes, it accesses sequence virtual addresses in the address space. If the page containing a virtual address is resident in main memory, the information at the virtual address is properly obtained or updated.

When a program references a page and then cannot find it in main memory, this creates what is called a " Page Fault". After a page fault has occurred it is required for the operating system to read in the required pages from the secondary, or virtual memory. Then the operating system must enter its new physical memory location in the page table, and then repeat the instructions (developer, web). If the operating system cannot complete the task the system will crash. This is also known as " Fatal Page Fault".

Address mapping device is the essential hardware element in virtual memory. It is located between the processor and the main memory. This device uses as page table to translate each virtual address in the address space, to a real address in main memory (Carr, 1984). The address-mapping device would interrupt execution if the page were missing. It would then signal the operating system for a page fault. Two activity indicators exist in the main memory frame, a use-bit, and a dirty-bit. The bits are in a separate memory and have one pair for each main memory frame. The used-bit indicates that the page is active and, most likely, should not be replaced. The dirty-bit indicates that the page must be moved to an auxiliary memory before it can be replaced (Carr, 1984). When a missing page is referenced the operating system creates the page table for each task. The system, then, selects a main memory frame to hold the missing page and performs an I/O operation to move the page from its slot to the frame. The frames that have not been allocated can be chosen. All the frames are physically identical except for its real address. This brings us to the page replacement algorithm. There are many page replacement algorithm policies, such as; LFU (Least Frequently Used), LRU (Least Recently Used), and Random. The most effective page replacement algorithm policy is the LRU. It will replace the memory slot being used the least amount of time.

A problem that can be created by virtual memory is when the overall supply of memory is running low then swapping may occur. If the memory is extremely low, the system may spend all its time swapping memory in and out of disk. This would leave little time to accomplish purposeful work. This problem is commonly referred to as disk " thrashing" (developer, web). The following paragraphs will discuss various operating systems and how they manage virtual memory.

The virtual memory manger of Apple computers operates invisibly to the applications and its user. Most applications do no need to know if virtual memory has been installed unless they have critical timing requirements, execute code at interrupt time, or perform debugging operations (Apple, web). A couple of requirements still exist for Apple systems to use virtual memory. First, the computer needs to be running system software 7. 0 or higher. Second, the computer must be equipped with an MMU (main memory unit) or PMMU co-processor. Apples 68040 and 68030 based machines have an MMU built into the central processing unit and are able to run virtual memory with no additional hardware. In Apple computers the user can also set the virtual memory through the control panel. This can be compared to the system in Windows where the user is allowed to change the size or disable virtual memory. When the virtual memory is activated the logical address space is greater than the physical memory. Many factors can create fluctuations in the size of the logical address space these include:

The addressing mode currently used by the Memory Manager

the amount of space available on a secondary storage device for use by the backing-store file

if 24-bit addressing is in operation, the number of NuBus expansion cards, if any, installed in the computer (Apple, web)

24-bit addressing allows the memory manager to expand the size of virtual memory to half of the actual RAM available to the system.

The original version of the Macintosh operating systems was not a user of virtual memory. A particular location in RAM could be accessed by its physical address. Therefore, there was no difference between the virtual and actual address space. Since both the hardware and software have been revolutionized so significantly, this forced the Apple operating system to separate the logical address space from the physical address space. This was accomplished by using the MMU coprocessor to map the logical addresses to their corresponding physical addresses (Apple, web).

In Macintosh IIc: with 8MB of physical RAM the physical memory appears to the CPU and to the NuBus expansion as two individual 4MB ranges. In addition, the operating system uses the MMU coprocessor to determine the physical address corresponding to the logical address (Apple, web).

In Windows NT, although the processes are used in general when restoring addresses in a process, no pages of physical memory are committed, and perhaps more importantly, no space is reserved in the page file for backing memory. Also reserving a range of addresses is no guarantee that at a later time there will be no physical memory available. Instead, it is simply saving a specific free address range until needed (NT, web).

The Virtual Allow function can be invoked to reserve a range of addresses using the following code:

```
/*Reserve a 10 MB range of addresses */

1pBase = VirtualAlloc (NULL,

10485760

MEM_RESERVE,

PAGE_NOACCESS) ;
```

Reserved addresses can only be used when memory is first committed to the addresses. Memory can be committed as little as one page at a time. The maximum amount of memory that can be committed is based solely on the maximum range of contiguous free or reserved addresses (NT, web). In the Windows NT virtual memory system, page tables are used to access physical pages of memory. Each page table is itself a page of memory (NT, web). To free virtual memory addresses have to be set as either reserved or committed and by using virtual free the operating system ca free the amount of space on the hard drive. Processes in Windows NT have a minimal set of pages called a working set that in order for the process to run properly, must be present in memory when running (NT, web). Windows NT assigns a default number of pages to a process to a process at start up and gradually turns that number to achieve a balanced/ optimum performance. It also keeps a check if the process has the needed pages in physical memory. In Windows NT the virtual memory management functions offer direct control over the virtual memory. It splits 2GB-user address space into regions of memory that are reserved, committed, or free virtual addresses. It also allows for applications to alter the state of pages in the virtual address space.

Virtual memory made its appearance in to Unix with the introduction of the VAX-11/780 in 1978, with its 32-bit architecture, 4 gigabytes address space, and hardware support for demand paging (Vahalia 96). The demand paging is responsible for separating both memory and the process address space in to fix size pages. Also Unix system is intelligent enough to anticipate the next transfer or anticipatory paging. The program size is free to grow and shrink as the demand rises or falls it can maximize its size up to 4 gigabyte for a 32-bit machine (Vahalia 96). But it must be noted since the size of the program can be real large, increased transfer between the virtual memory and the physical memory can slow down the system dramatically. The address space used by each Unix process is divided into a number of regions. Each region is a kernel structure used to represent a contiguous area of non-overlapping virtual addresses and is governed by its own set of attributes defining the region type and additional properties such as whether the region can be read from, written to or executed. The following are data structures that define the virtual memory area in Unix:

```
" Struct vm_area_struct {
```

/* parameters for virtual memory area*/

struct task_struct * vm_task;

unsigned long vm_start;

unsigned long vm_end;

pgprot_t vm_page_prot;

unsigned short vm_flags;

/*AVLtree for virtual memory area of process, sorted by addresses */"(Beck 96)

the code vm_task is the pointer pointing to the entry of the process table to which the area of memory is allocated. Vm_start and vm_end determine the beginning and the end of addresses in virtual memory. vm_page_prot safeguards characteristic of pages in the virtual memory. The system accessibility such as read access or read-write access is stored in the vm_flags.

Virtual memory for the most part has the same concept in all the operating system, what makes them different is the way they manage it. Operating system takes advantage of this phenomenal feature, which has changed the way computers are used forever.

Bibliography

1. About the Virtual Memory Management http://developer. apple. com/techpubs/mac/memory/memory-152. html.

2. Managing virtual memory in win 32 http://www. microsoft. com/win32dev/base/virtmm. htm.

3. Carr, Richard W. virtual memory management UMI research press, 1984.

4. Tanenbaum Andrew S. Structured computer organization Englewood cliff NJ: Prentice hall 1976.

5. Vahalia, Uresh UNIX Internal The New Frontier Upper saddle river NJ: Prentice hall, 1996.

6. Beck, M., Bohme H., Dziadzka M., Kunitz U., Magnus R., and Verworner D. LINUX Kernel Internals Harlow, England: Addison Wesley Longman limited 1996.