OPERATING SYSTEM

Lecture 4

Virtual Memory and Cache Memory

Dr Victoria Mukami

INTRODUCTION

This lecture looks at the concepts of virtual memory and cache memory. Specifically, hardware and control structures will be reviewed. Additionally, the lecture will focus on how virtual memory works on the Windows and Linux operating systems. Finally, a review of how cache memory works.

Learning objectives

By the end of this topic, you should be able to:

1. Understand various terminology associated with virtual memory
2. Review the inner working of virtual memory on specific operating systems
3. Understand how cache memory is utilized by the operating system.

OVERVIEW

So far, we have reviewed the various memory management systems that are used by operating systems. Specifically, we have looked at the early memory management systems and the more recently used memory management systems. We first review the various terminology associated with virtual memory.

Virtual Memory: A scheme that allows for secondary memory to be viewed and used like main memory [2]. Virtual memory is normally limited by the amount of secondary storage available as well as the addressing scheme by the computer.

Virtual Address: This is the address that a virtual memory location is assigned that allows for it to be accessed as though it is the main memory [2].

Address Space: These are the memory addresses that are available for a process [2]

VIRTUAL MEMORY

Virtual memory gives a user the illusion that the computer has more memory than it does. This takes care of the issues we have seen with previous memory allocation schemes. where jobs that required resources larger than the main memory could not be processed. With virtual memory, large jobs can be handled comfortably by the computer. Virtual memory was a result of pages being able to be allocated and deallocated to and from main memory to secondary storage [1]. Swapping pages in

and out of memory give the user the illusion that the entire job is stored within memory during processing [1].

Virtual Memory can be combined with Paging and Segmentation. We discuss the characteristics of each.

Virtual Memory with Paging

When combined with paging, virtual memory requires the use of a page map table. The programs similarly to paging are divided into the same size pages. VM with paging does not have any external fragmentation however, internal fragmentation still occurs within the page frame. Additionally, a free frame list needs to be maintained. With this scheme, all the pages do not have to be located within the main memory as they can be partly located within the secondary memory. The absolute address of the VM is calculated using a page number and displacement [1].

Virtual Memory with Segmentation

When combined with segmentation, virtual memory requires the use of a segment map table. The programs using this scheme are like segmentation and are divided into segments. As with segmentation due to dynamic allocation of memory, there is no internal fragmentation, however, there is external fragmentation in between memory frames. A list of free memory spaces needs to be maintained by the operating system. With this scheme, all the segments do not have to be located within the main memory as they can be partly located within the secondary memory. The absolute address of the VM is calculated using a segment number and displacement [1].

Virtual memory is a tool that allows for efficiency in multiprogramming environments. Normally, the CPU wastes time when resources are yet to be availed to allow for the processing of a job. In the case of VM, the CPU can move on to other jobs before the resources are availed [1]. There are several benefits and drawbacks to VM.

Benefits

- The first advantage of course is the fact that the job’s size is not restricted to the main memory size. This is a win, especially for large jobs.

- As with swapping, memory usage is efficient as only sections (pages or segments) of a job that are required for immediate processing are stored in memory [1].
- Multiprogramming is unlimited as there is either dynamic or static partitioning of memory and the inclusion of secondary memory [1].
- It allows for the sharing of code and data between programs and users [1].

Drawbacks

- The use of VM increases processor hardware costs. This is due to the use of the secondary memory acting as the main memory [1].
- There is increased use of computer resources due to the overhead of handling interrupts [1]
- Remember thrashing, VM increases software complexity to prevent thrashing [1].

WINDOWS VIRTUAL MEMORY MANAGEMENT

In the Windows operating system in this case the memory manager in charge of virtual memory oversees how memory is allocated and how paging is performed [2]. Page sizes within this system vary and range from 4Kb to 64Kb. Windows comes in either 32-bit or 42-bit platforms. 32-bit platforms allow for up to 4GB of virtual memory per process [2]. As seen in the various memory management schemes, half of this memory is reserved for the operating system functions. Within 64-bit systems, 8TB is available for use as VM within windows [2].

During paging, the space, either 2TB or 8TB is divided into equal size pages that can be brought into memory. The spaces are managed in a contiguous manner by the operating system. The virtual memory manager works with the following rules:

1. When there is plenty of memory, the VM manager allows the active processes to grow. Older pages are not swapped out during this stage.
2. When there is scarce memory, the VM manager will recover memory by removing pages that have not been active/used recently.
3. Windows will continuously keep watch for large programs that keep increasing the amount of memory they need. The operating system will then work to remove the least recently used processes.

LINUX MEMORY MANAGEMENT

Linux and Unix almost share similar characteristics when it comes to memory management. Page sizes within Linux depend on the Linux architecture but are normally 4KB. With virtual memory, the Linux operating system uses a three-level page table structure [2]. The **page directory** is found within the main memory and represents an active process. **Page middle directory** that represents one page within the page table. The **page table** represents one virtual page of a process.

Linux uses a buddy system to maintain the pages to and from memory. This increases the efficiency with which pages are written to and from memory. A kernel maintains a contiguous list of fixed-size page frames [2]. The Buddy algorithm is used during allocation and deallocation.

CACHE MEMORY

While cache memory is invisible to the operating system, it interacts with other hardware systems [2]. Virtual memory and cache memory work in a similar manner. Based on previous classes we have noted that there are three levels of cache memory used in today's computer systems.

Level 1: This is fast cache memory but can only hold small amounts of information.

Level 2: Much slower than level 1 cache memory but can hold larger amounts of information.

Level 3: Much slower than level 2 but holds a lot of information.

Figure 1 is an adapted version of how cache memory works [2].

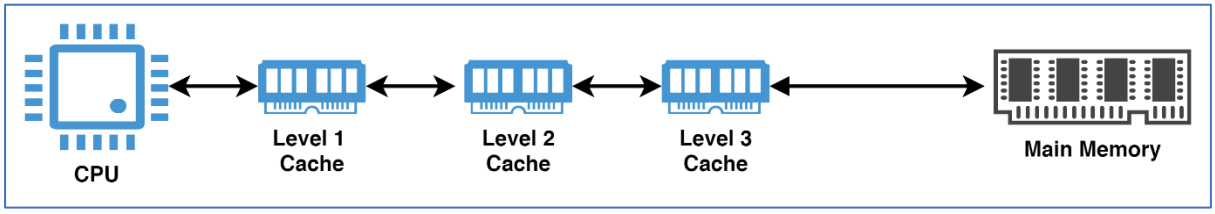

Figure 1: Cache memory levels interaction [2]

Cache memory uses small, fast, and expensive memory to supplement the main memory, especially when dealing with the CPU [1]. The cache is very fast as compared to the main memory up to ten times faster. Cache memory helps to lower the amount

of time needed to access different instructions. Figure 2 shows how data was transferred between the secondary memory and the main memory. Data transfer was still slow, hence necessitating the need for a middleman that was faster than the main memory. Frequently needed data or instructions are stored within the cache memory.

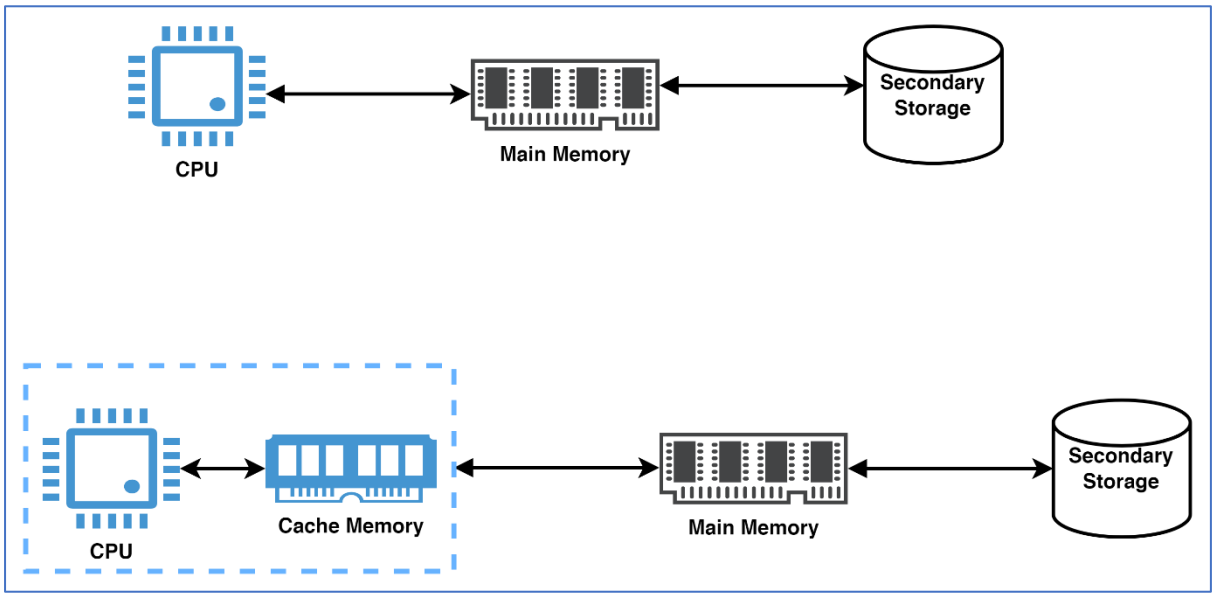

Figure 2: Cache interaction with memory (Adapted) [1]

The movement of data from the main memory to cache memory is like the previously discussed paging scheme. As shown previously, the memory is first divided into equal sizes. Cache memory in this regard is divided into equal size blocks known as slots [1]. After the CPU has requested data or instructions, they are transferred into cache memory from the main memory and placed into any of the free slots. Thereafter, anytime the CPU needs frequently used data, it is accessed from the cache memory and if the instructions are found within the cache the main memory is not accessed. Block replacement when cache memory slots are all busy is done similarly to the paging scheme.

The memory manager has several roles during memory management and the evolution of memory management schemes has been precipitated by previous schemes. Table 1 below shows a summary of all the learnt memory schemes and the issues created or solved [1].

Table 1: Memory allocation schemes comparison [1]

Scheme	Problem Solved	Problem Created
Single User Contiguous		• Job limited to memory size • CPU often idle
Fixed Partitions	Idle CPU Time	• Internal Fragmentation • Job limited to partition size
Dynamic Partitions	Internal Fragmentation	• External Fragmentation
Relocatable Dynamic Partitions	External Fragmentation	• Job limited to memory size • Compaction overhead
Paging	Need for compaction	• Memory needed for tables • Job limited to memory size • Internal Fragmentation
Demand Paging	Job limited to memory size Inefficient memory use	• Many tables • Thrashing • Overheads due to page interrupt
Segmentation	Internal Fragmentation	• Hard to manage variable-length segments • External fragmentation
Segmented/ demand paging	Segments not loaded in demand	• Overhead for table handling • Memory for page and segment tables

SUMMARY

This lecture reviewed concepts related to virtual memory and cache memory. Specifically, hardware and control structures were reviewed. Additionally, the lecture focussed on how virtual memory works on the Windows and Linux operating systems. Finally, a review of cache memory was done.

DISCUSSION TOPIC

Throughout the last three lectures, we have reviewed the types of memory management schemes. Operating systems have come a long way in managing memory. Do online web research and find what new methods of memory management currently exist. Review the pros and cons of each. Submit this to your instructor or present it in class.

REFERENCES

[1] McHoes, A., & Flynn, I., Understanding Operating Systems. Boston: Cengage Learning, 2018

[2] Stallings, W., Operating Systems: Internals and Design Principles. Harlow: Pearson Education Limited, 2018.