Virtual Memory

Eric McCreath

Introduction

Virtual memory is a technique that permits processes to be executed even when they are not completely in memory.

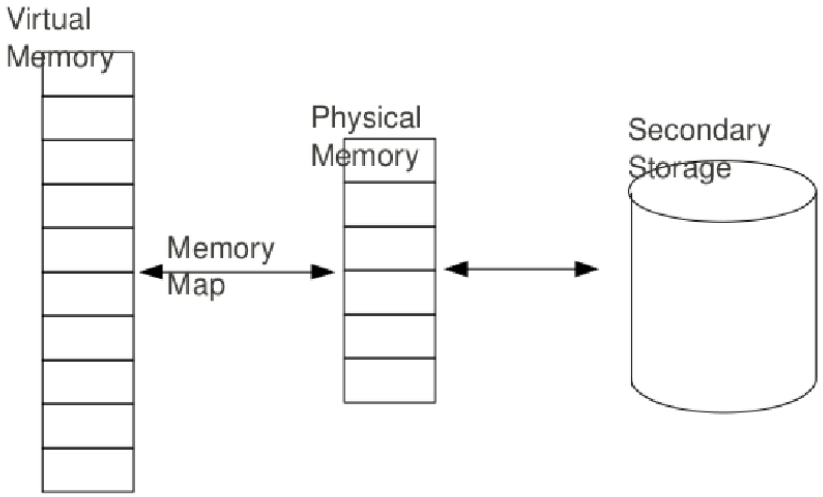
This has many advantages, including:

- programs can be larger than physical memory,
- virtual memory abstracts main memory into an extremely large logical storage area, and
- virtual memory increases the degree of multi-programming.

However, it is complex to implement and can dramatically decrease performance if it is used carelessly.

Introduction

The diagram below shows virtual memory larger than physical memory.



Demand Paging

Swapping is an approach that involves bringing an entire process from disk into main memory so it may be executed.

Demand-paging is a lazy "swapper" which brings into memory the pages of the process that are needed. The pager is concerned with the manipulation of the pages to and from disk.

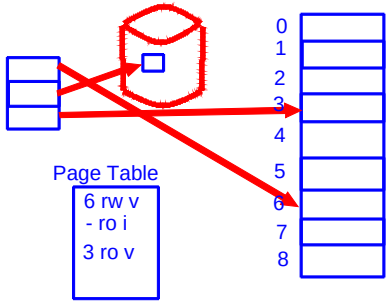
Hardware Support

If a logical address is accessed that is not in physical memory then the pager must bring this page into memory.

This requires some hardware support to distinguish between pages that are in memory and those that are on disk. A valid-invalid bit in the page table may be used to achieve this. If a page is invalid then either it is on disk or the page is not within the logical address space.

Demand-Paging

A page fault occurs when an invalid page is addressed.



Pure demand paging

One approach to paging is to only bring pages into memory when they are needed. This is called Pure demand paging.

The very first instruction would cause a page fault.

Demand paging can significantly decrease the performance of a computer system by greatly increasing the effective access time of memory.

Given ma is the memory access time. Let p be the probability of a page fault occurring. The effective access time is then:

$$\text{effective access time} = (1-p) \times ma + p \times \text{page fault time}$$

Swap Space

- Demand paging must manage the swap space. The swap space is generally faster than the file system, as file lookups and indirect allocation methods are not used.
- When a process is started, one approach is to copy the entire program into the swap space and then to demand page the pages from the swap space.
- Another approach is to page the program directly from the file system the first time a page is used, and then to use the swap space for the following page-faults.

Page Replacement

Once the main memory fills up a page must be swapped out to make room for any pages to be swapped in. This is known as page replacement.

This requires:

- Frame-allocation algorithm - How many frames do we allocate to each process?
- Page-replacement algorithm - How do we select the victim to be replaced?

Page-Replacement Algorithms

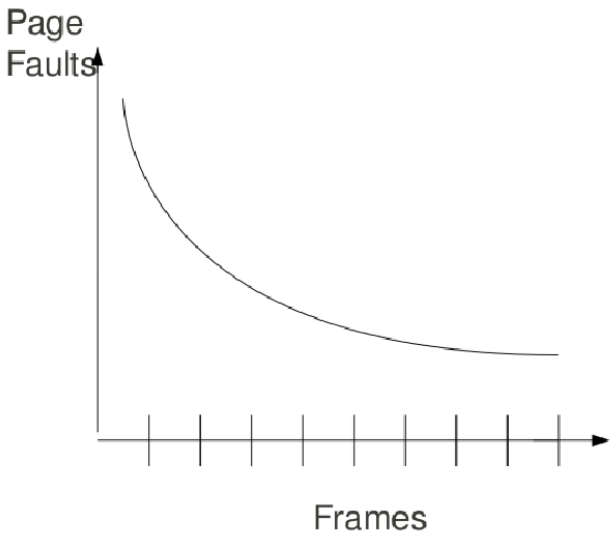
The goal of the page-replacement algorithm is to minimise the page-fault rate.

Different algorithms may be compared by computing the number of page faults on a particular reference string.

Given the overhead of a page fault, small improvements in the page replacement algorithm will greatly improve the performance of the entire system.

Page-Replacement Algorithms

Generally, increasing the number of frames reduces the number of page faults.



Page-repacement Algorithms

- The FIFO is the simplest page-replacement algorithm. When a page fault occurs and the page frames are full a victim must be selected. The FIFO algorithm selects the oldest frame (this is the frame that has been in memory the longest). This page is swapped out and the required page is swapped into its location.
- An optimal page-replacement algorithm exists and is called OPT or MIN. The approach is to simply replace the page that will not be used for the longest period of time.
- The page references that occurred in the recent past are good indicators of what page references will occur in the future. That is if a page has just been referenced it is likely that it will be referenced again. This gives rise to the least recently used (LRU) algorithm.

How do the three approaches compare on the following reference string (with 3 frames of memory available).

1 3 1 2 4 1 4 3 4 4 5 6 4

A pool of free frames may be maintained. When a page fault occurs, a victim is chosen as before. However, the required page is read into a free frame

from the pool of free frames. The process may continue once this is complete without waiting for the victim to be written out to the swap space. The victim may be copied to the swap space at the pages leisure and its frame will form part of the pool of free frames.