Review: Virtual Memory Management in the VAX/VMS Operating System

Robert Hoff

October 12, 2011

1 Paper Summary

The paper from 1982 describes the virtual memory management system of the VAX/VMS Operating System. It looks at the software components of the system, and the relationship to the underlying hardware, the VAX-11 family of minicomputers. The system aims to provide for different type of processes (real-time, time-shared and batch) to execute simultaneously, and to achieve this on a wide variety of hardware. At the time a greater variance of hardware systems were emerging, such as increased ranges in disk speed, memory size and CPU performance. The essential performance related operations of the system are highlighted and demonstrated, which include features related to the page tables, paging and swapping.

2 The Problem

The VAX-11 32 bit microprocessor family was introduced to replace the ageing PDP-11 16 bit systems, and the VAX/VMS Operating System was developed to work on the VAX instruction set architectures, to replace multiple operating systems that existed for the PDP-11. These developments were in focus when the mainstream computer market was shifting from mainframe to minicomputers, which had a more unpredictable composition of hardware. Demands for computers to support more multiprocessing activities were also increasing. The VAX/VMS was therefore roughly developed with the following specifications (i) to be able to run on minimal hardware, for example without specialised components such as fixed-head paging disks, and to not make too high demands on disk speeds or memory size. (ii) To support environments running multiple processes of different characteristics (iii) Not making sacrifices to efficiency

3 The Solution

To minimise demands on special hardware the process page tables were placed in the system virtual space. The translation of a process page virtual address would therefore require two accesses, one to the system page table, and then to the process page table. However the VAX-11 hardware provides a TLB so that multiple references to page tables are usually avoided.

A main innovation with the VAX/VMS is the introduction of the free and modified page lists. These lists act as temporary caches for pages that have been removed from a process' resident set. If the free or modified page lists become full the pages must either respectively be deleted or written back to disk. If however a process faults a page that is on either list the page is returned to the process' resident set. In addition to introducing the efficiency of returning pages at minimum cost, the existence of the free and modified page lists allow for the system to cluster pages together in either of three cases; when returning them to a resident set, deleting them, or writing them back to disk. The clustered segments are usually defined over contiguous pages with similar attributes.

Using additional page lists enables simpler replacement policies, in the VAX/VMS reference bits are avoided, and a simple FIFO replacement algorithm is used that only operates local to the executing process. It is shown in tests that with the introduction of additional page lists, the simple replacement policy would be almost as efficient as a LRU (Least Recently Used) replacement.

4 Evaluation

The unorthodox choices in the design, which include the organisation of the process page tables and the page replacement policy, where made because of the variety of the intended hardware. Although simplicity was retained, major mechanisms were introduced to enhance performance. These were the page caching, clustering, and process-local replacement.

5 My Opinion

I'm not convinced about the process-local page replacement policy, together with the size limits imposed on processes. It would seem likely that if a given process is allocated a range that is too narrow, then repeated page-faults would occur as a result of high page turnover.

Placing the page tables in the system virtual space I believe would place unnecessary stress on the TLB. If a given system offers hardware support it's uncertain if the VAX/VMS performs necessary checks to take advantage of such hardware.

It looks like the performance simulation given shows the use of caching and first-level FIFO against a system without caching and first-level LRU. It would be interesting to know if further performance enhancements could be achieved with more sophisticated replacement mechanisms, while retaining the caching.

6 Possible Questions

1. It would be useful to know in more detail the mechanism of the swapper and if considerations were taken to ensure it's efficiency, and possibly, the amount of time it spends compared to the pager. I was wondering for example if the swapper is limited to operating on entire processes, and whether *demand paging* may be a better solution?

2. Several memory functions, such as growing or shrinking a process' P0/P1 regions, are left to the operating system. It would be useful to know, especially since the VAX/VMS uses local-replacement, how carefully these functions would need to be controlled by the OS to ensure efficient performance, and whether some of these functions may be better delegated to the memory system?