Review: Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architechtures

Robert Hoff

October 12, 2011

1 Paper Summary

The paper from 1987 describes the virtual memory design for the CMU Mach Operating System. The main objective of the Mach memory system was to make it easily portable, by isolating a minimum amount of software that is hardware dependent, while not making sacrifices to efficieny or functionality. It is shown that the machine-dependent portion was small, relatively easy to implement, and did not negatively impact the overall performance in tests against a collection of less portable systems.

2 The Problem

At the time the variety of hardware arcitectures were increasing and operating system implementations where needed for each type. A possible solution to the demands of the variety is to increase the portability of a single operating system by achieving a separation of memory management and hardware support. The Mach system attempts to do this by retaining speed, and support for optimising features such as large virtual address spaces and memory sharing.

3 The Solution

The hardware dependent code is decoupled by a single code module that performs all the machine dependent mappings.s The seperation is achieved by the software abstraction on top of the physical memory and mappings, consisting of memory objects and message-passing functions. Physical memory is considered as a 'cache' for virtual memory. Each physical page entry may be simultaneously referenced by several lists. *Memory object list* All page entries associated with a given object are linked together in a memory object list to speed up object deallocation and virtual copy operations

Memory allocation queue are maintaned for free, reclaimable and allocated objects. *Object/offset hash bucket* allows fast lookup of physical pages at fault time. Byte offsets in memory objects allow Mach to use different page sizes.

4 Evaluation

Assesment on performance were made against several different architectures and measurements. It was shown that performance was good even with the isolation of hardware dependent parts, so the increased portability is not made against a significant performance cost.

By isolating the hardware dependent code, it makes it easier to perform studies that compare hardware performance

5 My Opinion

The performance comparisons are made against the 4.3bds that had limited support for virtual memory. It would have been interesting to know what the performance relations had been hardware specific systems that support the same functionality.

Today the emphasis on portability may be less relevant because we've moved towards greater standardisation of hardware.

It's indicated that the Mach kernel may perform better than a comparable UNIX platform. The reason for this was not clear as one would expect that the Mach might perform worse, since as a tradeoff for a generic design it may not embrace all features of the underlying hardware.

6 Possible Questions

1. when using the operating system across various hardware architectures, would it be possible to isolate the relative exectution times between the hardware dependent and independent parts? or if it would be possible to perform tests against hardware specific systems offering the same functionality?

2. It would be interesting to know in more about how relevant the comlexity of the shadow object chains are to normal operation, in what circumstances these arrise, and if this is a potential security issue? Based on this one might raise concerns on whether there may be better solutions for the copy-on-write memory management.