© 2014 IJIRT | Volume 1 Issue 6 | ISSN : 2349-6002

MACH KERNEL

Pranshu Sharma , Satish Anand Arjunan

I. INTRODUCTION

Mach is an operating system kernel developed at Carnegie Mellon University to support operating system research, primarily distributed and parallel computation. It is one of the earliest examples of a microkernel. Its derivatives are the basis of the modern operating system kernels in Mac OS X and GNU Hurd.

The project at Carnegie Mellon ran from 1985 to 1994, ending with Mach 3.0. Mach was developed as a replacement for the kernel in the BSD version of UNIX, so no new operating system would have to be designed around it. Today further experimental research on Mach appears to have ended, although Mach and its derivatives are in use in a number of commercial operating systems, such as NeXTSTEP and OPENSTEP, and most notably Mac OS X using the XNU operating system kernel which incorporates Mach as a major component. The Mach virtual memory management system was also adopted by the BSD developers at CSRG, and appears in modern BSD-derived UNIX systems, such as FreeBSD. Neither Mac OS X nor FreeBSD maintain the microkernel structure pioneered in Mach, although Mac OS X continues to offer microkernel inter-process communication and control primitives for use directly by applications.

Mach is the logical successor to Carnegie Mellon's Accent kernel. The lead developer on the Mach project, Richard Rashid, has been working at Microsoft since 1991 in various top-level positions revolving around the Microsoft Research division. Another of the original Mach developers, Avie Tevanian, was formerly head of software at NeXT, then Chief Software Technology Officer at Apple Computer until March 2006.

II. DEVELOPMENT

Mach was initially hosted as additional code written directly into the existing 4.2BSD kernel, allowing the team to work on the system long before it was complete. Work started with the already functional Accent IPC/port system, and moved on to the other key portions of the OS, tasks and threads and virtual memory. As portions were completed various parts of the BSD system were re-written to call into Mach, and a change to 4.3BSD was also made during this process.

By 1986 the system was complete to the point of being able to run on its own on the DEC VAX. Although doing little of practical value, the goal of making a microkernel was realized. This was soon followed by versions on the IBM PC/RT and for Sun Microsystems 68030-based workstations, proving the system's portability. By 1987 the list included the Encore Multimax and Sequent Balance machines, testing Mach's ability to run on multiprocessor systems. A public Release 1 was made that year, and Release 2 followed the next year.

Throughout this time the promise of a "true" microkernel was not yet being delivered. These early Mach versions included the majority of 4.3BSD in the kernel, a system known as POE Server, resulting in a kernel that was actually larger than the UNIX it was based on. The idea, however, was to move the UNIX layer out of the kernel into user-space, where it could be more easily worked on and even replaced outright. Unfortunately performance proved to be a major problem, and a number of architectural changes were made in order to solve this problem. Unwieldy UNIX licensing issues were also plaguing researchers, so this early effort to provide a non-licensed UNIX-like system environment continued to find use, well into the further development of Mach.

The resulting Mach 3 was released in 1990, and generated intense interest. A small team had built Mach and ported it to a number of platforms, including complex multiprocessor systems which were causing serious problems for older-style kernels. This generated considerable interest in the commercial market, where a number of companies were in the midst of considering changing hardware platforms. If the existing system could be ported to run on Mach, it would seem it would then be easy to change the platform underneath.

Mach received a major boost in visibility when the Open Software Foundation (OSF) announced they would be hosting future versions of OSF/1 on Mach 2.5, and were investigating Mach 3 as well. Mach 2.5 was also selected for the NeXTSTEP system and a number of commercial multiprocessor vendors. Mach 3 led to a number of efforts to port other operating systems parts for the microkernel, including IBM's Workplace OS and several efforts by Apple Computer to build a cross-platform version of the Mac OS.

III. PERFORMANCE PROBLEMS

Mach was originally intended to be a replacement for classical monolithic UNIX, and for this reason contained many UNIX-like ideas. For instance, Mach used a permissioning and security system patterned on UNIX's file system. Since the kernel was privileged (running in kernel-space) over other OS servers and software, it was possible for malfunctioning or malicious programs to send it commands that would cause damage to the system, and for this reason the kernel checked every message for validity. Additionally most of the operating system functionality was to be located in user-space programs, so this meant there needed to be some way for the kernel to grant these programs additional privileges, to operate on hardware for instance.

Some of Mach's more esoteric features were also based on this same IPC mechanism. For instance, Mach was able to support multi-processor machines with ease. In a traditional kernel extensive work needs to be carried out to make it reentrant or interruptible, as programs running on different processors could call into the kernel at the same time. Under Mach, the bits of the operating system are isolated in servers, which are able to run, like any other program, on any processor. Although in theory the Mach kernel would also have to be reentrant, in practice this isn't an issue because its response times are so fast it can simply wait and serve requests in turn. Mach also included a server that could forward messages not just between programs, but even over the network, which was an area of intense development in the late 1980s and early 1990s.

Unfortunately, the use of IPC for almost all tasks turned out to have serious performance impact. Benchmarks on 1997 hardware showed that Mach 3.0-based UNIX single-server implementations were about 50% slower than native UNIX.

Studies showed the vast majority of this performance hit, 73% by one measure, was due to the overhead of the IPC. And this was measured on a system with a single large server providing the operating system; breaking the operating system down further into smaller servers would only make the problem worse. It appeared the goal of a collection-of-servers was simply not possible.

Many attempts were made to improve the performance of Mach and Mach-like microkernels, but by the mid-1990s much of the early intense interest had died. The concept of an operating system based on IPC appeared to be dead, the idea itself flawed.

In fact, further study of the exact nature of the performance problems turned up a number of interesting facts. One was that the IPC itself was not the problem: there was some overhead associated with the memory mapping needed to support it, but this added only a small amount of time to making a call. The rest, 80% of the time being spent, was due to additional tasks the kernel was running on the messages. Primary among these was the port rights checking and message validity. In benchmarks on an 486DX-50, a standard UNIX system call took an average of 21μs to complete, while the equivalent operation with Mach IPC averaged 114μs. Only 18μs of this was hardware related; the rest was the Mach kernel running various routines on the message. Given a syscall that does nothing, a full round-trip under BSD would require about 40μs, whereas on a user-space Mach system it would take just under 500μs.

When Mach was first being seriously used in the 2.x versions, performance was slower than traditional monolithic operating systems, perhaps as much as 25%. This cost was not considered particularly worrying, however, because the system was also offering multi-processor support and easy portability. Many felt this was an expected and acceptable cost to pay. When Mach 3 attempted to move most of the operating system into user-space, the overhead became higher still: benchmarks between Mach and Ultrix on a MIPS R3000 showed a performance hit as great as 67% on some workloads.

For example, getting the system time involves an IPC call to the user-space server maintaining system clock. The caller first traps into the kernel, causing a context switch and memory mapping. The kernel then checks that the caller has required access rights and that the message is valid. If it does, there is another context switch and memory mapping to complete the call into the user-space server. The process must then be repeated to return the results, adding up to a total of four context switches and memory mappings, plus two message verifications. This overhead rapidly compounds with more complex services, where there are often code paths passing through many servers.

This was not the only source of performance problems. Another centered on the problems of trying to handle memory properly when physical memory ran low and paging had to occur. In the traditional monolithic operating systems the authors had direct experience with which parts of the kernel called which others, allowing them to fine tune their pager to avoid paging out code that was about to be used. Under Mach this wasn't possible because the kernel had no real idea what the operating system consisted of. Instead they had to use a single one-size-fits-all solution that added to the performance problems. Mach 3 attempted to address this problem by providing a simple pager, relying on user-space pagers for better specialization. But this turned out to have little effect. In practice, any benefits it had were wiped out by the expensive IPC needed to call it in.

Other performance problems were related to Mach's support for multiprocessor systems. From the mid-1980s to the early 1990s, commodity CPUs grew in performance at a rate of about 60% a year, but the speed of memory access grew at only 7% a year. This meant that the cost of accessing memory grew tremendously over this period, and since Mach was based on mapping memory around between programs, any "cache miss" made IPC calls slow.

Regardless of the advantages of the Mach approach, these sorts of real-world performance hits were simply not acceptable. As other teams found the same sorts of results, the early Mach enthusiasm quickly disappeared. After a short time many in the development community seemed to conclude that the entire concept of using IPC as the basis of an operating system was inherently flawed.

IV. OPERATING SYSTEMS AND KERNELS BASED ON MACH

- GNU/Hurd
- Lites
- MkLinux
- mtXinu
- MachTen
- MacMach
- NeXTSTEP
- OSF/1
- Workplace OS
- UNICOS MAX
- Kylin
- XNU and Darwin, the basis of Mac OS X and IOS

REFERENCES

[1] Apple Inc. (February 26, 2013), Kernel and Device Drivers Layer, Apple Inc. (February 26, 2013), Mach Overview

^[2] Al Saracevic (March 27, 2006). "Adios Avie". The Technology Chronicles. Retrieved 23 January 2010.

[3] Singh, Amit (2006-07-28). "A Technical History of Apple's Operating Systems". osxbook.com. p. 103. Retrieved 18 March 2011.

[4] Tevanian, Avadis; Rashid, Richard F.; Golub, David B.; Black, David L.; Cooper, Eric; Young, Michael W. (1987). "Mach Threads and the Unix Kernel: The Battle for Control". "Proceedings of the USENIX Summer Conference, USENIX Association". pp. 185–197. CiteSeerX: 10.1.1.41.3458.

[5] Accetta, Mike; Baron, Robert; Bolosky, William; Golub, David; Rashid, Richard; Tevanian, Avadis; Young, Michael (1986). "Mach: A New Kernel Foundation for UNIX Development". "Technical Conference - USENIX".

[6] M. Condict, D. Bolinger, E. McManus, D. Mitchell, S. Lewontin (April 1994). "Microkernel modularity with integrated kernel performance". Technical report, OSF Research Institute, Cambridge, MA.

[7] Hermann Härtig, Michael Hohmuth, Jochen Liedtke, Sebastian Schönberg, Jean Wolter (October 1997). "The performance of μ-kernel-based systems". Proceedings of the 16th ACM symposium on Operating systems principles (SOSP), Saint-Malo, France 31 (5): 67. doi:10.1145/269005.266660. ISBN 0-89791-916-5. url2

[8] www.princeton.edu

[9] Jochen Liedtke (1993). "Improving IPC by Kernel Design". Proceedings of the 14th ACM Symposium on Operating System Principles (SOSP). ISBN 0-89791-632-8.

[10] Chen, J B; Bershad, B N (1993). "The impact of operating system structure on memory system performance". ACM SIGOPS Operating Systems Review (Association for Computing Machinery) 27 (5): 133. CiteSeerX: 10.1.1.52.4651.