The Integration of Virtual Memory Management and Interprocess Communication in Accent

[Abstract]

Robert Fitzgerald and Richard F. Rashid
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

All communication-oriented operating systems share the problem of getting data from one process to another. System designers have traditionally chosen one of two alternatives:

1. processes pass data by reference
2. processes exchange messages to pass data by value

By-value message systems typically require that message data be physically copied. Not surprisingly, data copying costs can dominate the performance of by-value message systems [2]. Such systems often limit the maximum size of a message, forcing large data transfers to be performed in several message operations [3, 6].

In systems that allow by-reference sharing of memory, processes may either share access to specific memory areas or entire address spaces. Messages are used only for synchronization and to transfer small amounts of data, such as pointers to shared memory. Communication between processes within a THOTH team [3] is an example of this approach.

By-reference sharing of data is much cheaper than copying for large data transfers on a single machine, but can seriously compromise system reliability and security. Several capability-based systems [4, 5, 9] have partially addressed this problem by passing memory access capabilities in messages. However, these systems do not address the problems of unintended or unsynchronized access to shared data. It is also difficult and expensive to extend a by-reference memory access scheme transparently into a network environment [8].

In 1981, we began to implement Accent, a communication-oriented operating system kernel designed to support the needs of a large network of personal computers. One of the Accent design goals was that its communication abstractions be transparently extensible into the network environment. We therefore chose to pass all data between processes by-value in messages. At the same time, experience with previous operating systems, notably Rochester's RIG system [6], led us to seek an alternative to data copying for large messages.

Our approach was to combine virtual memory management and interprocess communication in such a way that large data transfers could use memory mapping techniques rather than data copying. By-value semantics are preserved by transferring large amounts of message data with copy-on-write memory mapping, so that both the sending and receiving process have their own logical (if not disjoint physical) copy of the data.

Our hypothesis was that copy-on-write data would, in fact, seldom be written. By postponing data copy operations until they are actually necessary, we hoped to provide the cost advantages of by-reference memory mapping for interprocess communication with the clean semantics of by-value data copying. Lazy evaluation is also heavily used in the management of process maps and allocation of virtual memory backing store.

It has now been more than four years since we began to implement Accent as part of the CMU SPICE project. Accent[1] is now (September 1985) running on a network of approximately 200 personal computers at CMU and is marketed commercially by PERQ Systems Corp. and Advent Ltd. with an installed base of over 1000 systems.

In addition to network operating system functions such as distributed process and file management, window management and mail systems, several applications have been built using Accent's primitives. These include research systems for distributed signal processing, distributed speech understanding and distributed transaction processing. Four separate programming environments have been built -- CommonLisp, Pascal, C and Ada -- including language support for an object-oriented remote procedure call facility. A commercial version of UNIX System V has even been built as an application on top of the Accent kernel.

[1]Accent is a trademark of Carnegie-Mellon University.

The successful use of Accent for a wide variety of distributed applications at CMU and elsewhere has shown that interprocess communication and virtual memory management can indeed be combined to form workable primitives for the design and implementation of a network operating system. In order to judge the effectiveness of the design and implementation of Accent's communication abstractions, we measured its performance both on a series of message-oriented benchmarks and in normal operation.

Our measurements have demonstrated that these mechanisms can also be used to deliver single machine performance comparable to that of more traditional operating system designs. More than 61 percent of total time during a large-scale system generation task was delivered to processes in user state. The performance of kernel-intensive process creation and destruction operations is comparable to that of Unix4.1bsd on a VAX 11/780, after normalizing for differences in processor speed. Accent file reading performance is directly comparable to that of Unix4.1bsd.

Our measurements also confirm our hypothesis that the cost of copy-on-write memory management is nearly identical to that of by-reference memory mapping. The overall contribution of copy-on-write faulting to total system costs is extremely small. Less than 0.01 percent of total time during a system generation task was spent handling copy-on-write faults.

Lazy evaluation of memory map and backing store operations proved to be valuable. Of the physical pages for which allocation of backing store was postponed during the system generation task, fewer than 1 percent were ultimately recorded in process maps and backed on disk.

We found that in Accent, unlike more traditional message systems, the cost of simple message passing was much less important than the cost of virtual memory operations. These costs were dramatically apparent in our measurements of the system generation task and of process creation and destruction. Ironically, far more care was taken in the Accent implementation to streamline simple IPC operations than to minimize virtual memory management costs.

The basic design of the Accent virtual memory system appears sound. Our measurements show that the costs of manipulating the Accent process map data structure to allocate, free and copy mapped regions are fundamentally small and grow slowly with the size of the affected memory area.

Unfortunately, we found that the cost of actually taking a fault (3-4 milliseconds) or remapping a physical page of memory (800 microseconds in our original implementation) can easily dominate the costs of any process map manipulations. We have recently addressed these costs by moving some of the most expensive operations into microcode. Accent uses the flexibility of the PERQ's writable control store to overcome its speed deficiencies in the same way that other systems might use assembly language.

The effects of page size are important. Most system costs depend more strongly on the number of pages in a region than on the number of bytes in it. For largely historical reasons, Accent uses a 512-byte page. This small page size causes significantly more remapping of physical memory and more faulting operations than would occur with larger pages and reduces the effectiveness of address translation caches by reducing the size of the address range covered by a single cache entry. It also dramatically increases the costs of kernel data structures. The small disk page size often implies a large overhead to transfer a small amount of data. Experience with Unix systems [1, 7] indicates that the benefits of a larger page size would probably outweigh the costs of increased internal fragmentation.

Overall, the Accent implementation has satisfied its original goals. It provides an existence proof that a communication kernel with a few basic primitives can provide effective support for a large body of software. It has also demonstrated that a usable system can be built with its memory management and interprocess communication primitives and that these primitives can be implemented efficiently.

References

1. Babaoglu, O. and W. Joy. Converting a Swap-Based System to do Paging in an Architecture Lacking Page-Referenced Bits. In *Proc. 8th SOSP*, ACM, December, 1981, pp. 78-86.
2. Ball, J.E., E. Burke, I. Gertner, K.A. Lantz and R.F. Rashid. Perspectives on Message-Based Distributed Computing. In *Proc. 1979 Networking Symposium*, IEEE, December, 1979, pp. 46-51.
3. Cheriton, D.R., M.A. Malcolm, L.S. Melen and G.R. Sager. "Thoth, a Portable Real-Time Operating System". *CACM 22*, 2 (February 1979), 105-115.
4. Kahn, K.C. et al. iMAX: A Multiprocessor Operating System for an Object-Based Computer. In *Proc. 8th SOSP*, ACM, December, 1981, pp. 127-136.
5. Jones, A.K., R.J. Chansler, I.E. Durham, K. Schwans and S. Vegdahl. StarOS, a Multiprocessor Operating System for the Support of Task Forces. In *Proc. 7th SOSP*, ACM, December, 1979, pp. 117-129.
6. Lantz, K.A., K.D. Gradischnig, J.A. Feldman and R.F. Rashid. "Rochester's Intelligent Gateway". *Computer 15*, 10 (October 1982), 54-68.
7. McKusick, M.K., W.N. Joy, S.L. Leach and R.S. Fabry. "A Fast File System for UNIX". *ACM Transactions on Computer Systems 2*, 3 (August 1984), 181-197 .
8. Spector, A.Z. *Multiprocessing Architectures for Local Computer Networks.* Ph.D. Thesis, Stanford, 1981.
9. Wulf, W.A., R. Levin and S.P. Harbison. *Hydra/C.mmp: An Experimental Computer System.* McGraw-Hill, 1981.