System Software Support for High Performance Servers

Gregory R. Ganger, Héctor Briceño and M. Frans Kaashoek
M.I.T. Laboratory for Computer Science
{ganger,hbriceno,kaashoek}@lcs.mit.edu

To satisfy increasing demand for high performance distributed computing, system implementors must be able to construct a variety of server applications (e.g., HTTP, NFS, FTP, etc.) that can support large numbers of active clients. Currently, most servers are run on general-purpose operating systems (OSs), such as UNIX or Windows/NT, using high-level interfaces such as sockets and read/write file systems. Unfortunately, while the appropriate resource abstractions greatly simplify the construction and improve the portability of applications, the abstractions found in current OSs are a poor match to the needs of high performance server applications. As a result, software overheads and inefficient use of resources prevent these servers from delivering the full performance of the underlying hardware (in particular, the network and disks).

To address this short-coming, we are re-examining the abstractions needed for easy construction of portable server applications that deliver hardware performance [Kaashoek et al., 1996]. We believe that the right abstractions will allow server applications to: (1) integrate the performance critical aspects of the management of different hardware resources (e.g., main memory, disk and network), (2) manipulate low-level resource management details in a high-level and portable manner, (3) note and react quickly to relevant I/O events (e.g., network packet arrivals), and (4) exploit domain-specific knowledge to specialize policies.

Based on these desiderata, we are designing and implementing several abstractions that can be parameterized and combined in server-specific ways. As one example of applying the above precepts, our TCP/IP abstraction allows applications to specify that a particular transfer is the last for the connection (this information can be used to reduce the number of network packets sent) and to provide precomputed checksums for the data being transmitted. As a second example, our file system abstraction provides non-blocking interfaces for all operations, allows applications to store some of their own information in the inode, and allows applications to indicate that a particular set of files (e.g., an HTML document and any included images) should be treated as a unit for disk allocation, caching and prefetching purposes.

A third abstraction, a combined file cache/TCP retransmission pool, integrates these two performance critical functions to eliminate redundant memory copies. It pins all blocks that have been transmitted but not acknowledged and uses copy-on-write to deal with changes to file blocks. Combined with precomputed checksums, this abstraction enables data to be moved from the disk (or cache) to the network without being touched by the CPU. As a result, this organization eliminates many expensive cache misses, increasing data throughput and reducing CPU overhead.

We are implementing the new abstractions as libraries on an exokernel-based system [Engler et al., 1995], because the addition of new system software abstractions is greatly simplified by an extensible operating system. Many of the same abstractions could be added to conventional OSs, such as UNIX and Windows/NT, or used to simplify the development of dedicated server systems like Network Appliance's products [Hitz, 1995].

The first server built using our new abstractions is a HTTP server called Cheetah. Measurements on a DECstation 5000/125 show that Cheetah running on the Aegis exokernel can support up to an order of magnitude more client activity than either the Harvest cache or the NCSA 1.4 server running on Ultrix (e.g., 700 requests/second versus 80 for 100 Byte documents and 96 requests/second versus 42 for 10 KB documents over a 10Mbit/second ethernet). We are in the process of building more server applications (e.g., NFS and FTP) with the new abstractions, to identify how well they reduce development costs and improve performance in other client/server environments.

Inappropriate resource abstractions can severely restrict performance. By developing high-level abstractions that better fit the computing model of server applications, we hope to realize significant performance improvements with no increase in development complexity or reduction in portability.

References

[Engler et al., 1995] Engler, D., Kaashoek, M. F., and O'Toole, J. (1995). Exokernel: an operating system architecture for application-specific resource management. In *15th ACM SOSP*, pages 251–266.

[Hitz, 1995] Hitz, D. (1995). An NFS file server appliance. Technical Report 3001, Network Applicance Corporation.

[Kaashoek et al., 1996] Kaashoek, M., Engler, D., Wallach, D., and Ganger, G. (1996). Server operating systems. In *SIGOPS European Workshop*, pages 141–148.