High-Capacity Internet Middleware: Internet Caching System Architectural Overview

Gary Tomlinson
Novell, Inc.
122 E. 1700 S.
Provo, UT 84606 USA
1-801-861-7021
garyt@novell.com

Drew Major
Novell, Inc.
122 E. 1700 S.
Provo, UT 84606 USA
1-801-861-2540
rdmajor@novell.com

Ron Lee
Novell, Inc.
122 E. 1700 S.
Provo, UT 84606 USA
1-801-861-7555
rlee@novell.com

ABSTRACT

Previous studies measuring the performance of general-purpose operating systems running large-scale Internet server applications, such as proxy caches, have identified design deficiencies that contribute to lower than expected performance and scalability. This paper introduces a high-capacity proxy cache service built upon a specialized operating system designed to efficiently support large-scale Internet middleware. It suggests that specialized operating systems can better meet the needs of these services than can their general-purpose counterparts. It concludes with the measured performance and scalability of this proxy cache service.

1. INTRODUCTION

Previous studies [2,5,25] have shown general-purpose operating systems lack key abstractions necessary for large-scale Internet server applications. According to one study, "The performance of Internet server applications on a general-purpose operating system is often dismayingly lower than what one would expect from the underlying hardware [5]". Several of these studies [2,5] have modeled and implemented operating system extensions designed to alleviate these deficiencies.

Our microkernel [21] provides the execution semantics and interfaces necessary to reduce the observed impedance mismatches between the design assumptions of general-purpose operating systems and the requirements of Internet server applications. This microkernel provides large-scale resource provisioning for execution contexts, network connections, and persistent storage objects, along with innovative semantics for context scheduling, event notification, and I/O transport.

By coupling the key abstractions provided by this microkernel with well-known Finite State Machine processing models, Novell has developed Internet Caching System (ICS) [32], a fast and highly scalable proxy cache architecture implemented on our microkernel technology.

This paper outlines the characteristics of the microkernel and its support systems that together constitute a middleware platform. It presents the core design model of the proxy cache service with respect to the underlying platform. It then presents a series of performance measurements to demonstrate the performance and scalability of the ICS service.

2. OPERATING SYSTEM PLATFORM

The operating system platform used to host the ICS service has its roots in NetWare [20], a specialized operating system similar in concept to Exokernel [18] and Rialto [17]. This platform utilizes a fundamentally different design center from that typically found in general-purpose operating systems such as Unix [36].

First, the kernel and the processes that use it are not rigidly separated. All services and applications are implemented as loadable modules that extend the system. Loadable modules are given a resource context to which all of their resources are attached, including those provided by the operating system such as memory, interrupts, I/O end points, schedulable entities, event messages, and others.

Second, loadable modules may run in either the supervisor memory or in protected user memory. Rudimentary support is provided to trap and quarantine faulty supervisor memory loadable modules, while complete fault containment is provided for user memory modules.

Third, handles are used for resource end points rather than file descriptors. Whereas a typical Unix system can support a few thousand file descriptors per process, this system has no practical limit imposed. It is constrained primarily by memory and the algorithms used by the I/O components.

Figure 1 is a block diagram of the platform.

2.1 The Microkernel

The microkernel provides services that are roughly equivalent to those found in other systems of the same genre [9,13,14]. We will discuss the fundemental services of the microkernel as utilized by ICS.

2.1.1 Process Model

Loadable modules are somewhat analogous to Unix processes, but differ in several fundamental aspects: (1) their threads are those provided directly by the microkernel; (2) they can be directly linked with other modules including the microkernel itself; and (3) their

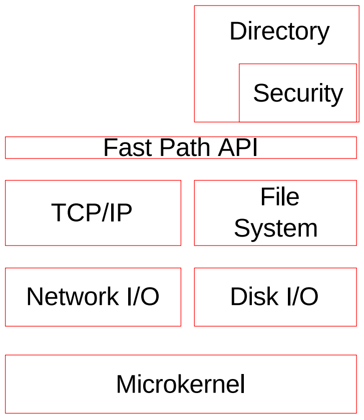

Figure 1. Platform Block Diagram.

execution contexts, known as *tasks*, may be bound either to threads or to "work objects." Any number of tasks can be associated with a loadable module. In the case of the ICS proxy cache service, there can be tens of thousands of tasks within one loadable module.

A task bound to a thread uses the thread's stack to maintain state. A task bound to a work object must maintain its own self encapsulated persistent state. In turn, a work object shares a pool of threads bound to its work queue with other work objects on the queue..

2.1.2 Scheduler

Support is provided for non-preemptive threads, preemptive threads, symmetric multiprocessing, and processor affinity.

There are two classes of work objects: fast-work-to-do and work-to-do, each with one run queue per processor. Work objects placed on fast-work-to-do queues run in FIFO order on the processor's current thread just prior to its thread switch. These work objects must be programmed to never block.

Each work-to-do run queue has a pool of dedicated threads. Work objects placed on these queues run in FIFO order on threads servicing that queue. These work objects may block, but must do so sparingly to avoid stalling the queue's thread pool.

2.1.3 Event Messages

The event model is based on an "up-call" model described by Clark [8]. This is a mechanism where a lower layer of a system can signal an event to a higher layer in the system by means of a procedure call known as an Event Service Routine (ESR) or call back routine. Event messages are used primarily in service interfaces; system resource advisories including volume mounts, low memory, etc.; and exception handling.

2.1.4 Inter-Process Communication

The inter-process communication (IPC) model is organized around the core linking facilities provided by the loader. In the case of intra-address space resolution, a direct link is established representing a NULL RPC between provider and consumer. In the case of inter-address space resolution, RPC stubs are generated in behalf of both the provider (server stub) and consumer (client stub). The inter-address space RPC mechanism uses a call gate and provides the necessary marshalling of parameters.

2.2 Platform Services

The microkernel may be the nucleus of the system, but it alone doesn't constitute a platform. The following standard services are provided to establish the middleware platform.

- *Fast Path API* is the service model to which all platform services subscribe. It differs substantially from the file and socket system call interfaces provided by Unix. First, it is implemented directly upon the IPC and event message systems provided by the microkernel. Second, it is completely asynchronous with event completion. Third, it supports vectored, direct unified I/O buffering [22] (this has been a standard part of shipping product for 11 years, in contrast to recent papers [3,35,37] on the subject). Finally, it provides resource handles rather than file descriptors.
- *Network I/O Service* provides network access to a comprehensive set of LAN and WAN topologies, including Ethernet, Fast Ethernet, Gigabit Ethernet, FDDI, Token Ring, X.25, Frame Relay, ATM, and others.
- *TCP/IP Service* provides both TCP and UDP transport to applications and services. IP provides additional functionality for routing, transparent proxy routing, network address translation, and IPSEC secure channels. Extended policy management is provided which allows the application to control packet merging and transmission, checksum caching and in the case of intelligent adapters, on-adapter DMA buffer transfers and optional on-adapter checksum processing.
- *Disk I/O Service* provides access to a comprehensive set of mass storage media, including IDE, SCSI, FireWire, USB, and others.
- *File Service* provides comprehensive file services, including support for multiple volumes, multiple spindles, and multiple name spaces. Intrinsically supported name spaces include Microsoft DOS, Microsoft Windows 98, Microsoft Windows NT, Apple Macintosh, and Berkeley Unix.
- *Directory/Security Service* provides authentication, configuration, system policy, and system management services. Support is provided for X.509 public key infrastructure [15], schema-based policy frameworks [11,16] and LDAP [40]. Objects are organized in hierarchical containers, under a unified access control model. Replication of the directory database is also supported.

3. ICS SERVICE MODEL

After investigating both multi-threaded and event-driven server models, we came to the same conclusions as previous studies [5] that thread or process per connection servers such as NCSA [27] and Apache [4] are less scalable than event-driven servers such as Harvest [7] and Squid [28].

3.1 Execution Model

We decided upon a hybrid execution model that uses both thread tasks for monitor operations and work object tasks for the event-driven proxy cache service finite state machines. All of the tasks are associated with a single loadable module. Figure 2 illustrates the execution contexts employed by ICS.

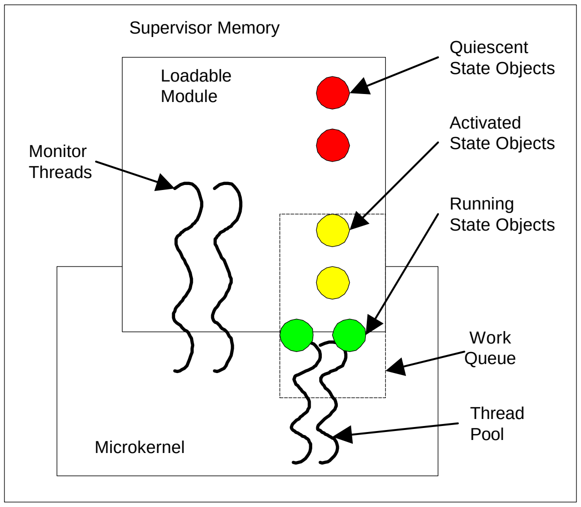

Figure 2. Execution Contexts.

3.1.1 Monitor Threads

These tasks do not need to run frequently, nor are very many needed to support the system. They may block and yield as necessary to complete their work. We chose tasks bound to threads for use with monitor activities in the system, such as configuration, system console, web console, and SNMP agents.

3.1.2 Finite State Machines

These tasks need to run in either real-time or pseudo real-time. A great number of them, sometimes many thousands, run concurrently. They are short lived, existing for the duration of a single transaction. We chose to build them as work object tasks based on the following design principles: (1) asynchronous event driven processing; (2) fast-work-to-do used to shuttle activation events to quiescent state objects; (3) work-to-do used for state processing; (4) avoid blocking; and (5) use multiprocessing for processor bound activities.

There are three execution context states:

- *Quiescent State*. Tasks waiting for an event are said to be in the quiescent state. They hold no execution context, are associated with work objects, and are not scheduled until an activation event occurs.
- *Activated State*. Tasks that are scheduled but not running are said to be in the activated state. They will be run in FIFO order by the next available work pool thread.
- *Running State*. Tasks bound to a thread are said to be in the running state. They hold the execution context of the thread they are running on, must avoid blocking, and must either terminate or transition to a new state.

3.2 Transaction Service Components

Proxy components are comprised of a series of finite state machines linked in chains to process requests. Figure 3 illustrates the ICS component architecture.

The *Connection Manager* provides network communication management for the client and server proxies. Substantial capability is added to the network service provided by TCP/IP, including: (1) data transfer between file system cache buffers and network channels; (2) flow control policy management of TCP; (3) network event shuttling; (4) multi-homed server load balancing and availability monitoring; and (5) SOCKS and SSL channel processing.

The *Object Cache Manager* manages caching policies and the RAM cache. It acts as a broker, coordinating the RAM cache, object store, and client proxies as necessary to service cache requests made by server proxies.

The *Cache Object Store (COS)* manages persistent storage of cached objects. It is responsible for migrating objects in the RAM cache to and from disk. It operates independent of the platform standard file system, directly utilizing the disk I/O system. This component is a recent addition to the system; earlier versions used the standard file system for persistent storage.

Previous studies measuring the performance of web proxy caches have identified the persistent storage system as a principal bottleneck of performance and scalability [1,6,10]. This is principally due to poor spatial locality of file system layout and temporal locality of the cache working set, which classic file systems were not designed to meet. COS is specifically designed for the operational requirements of a temporal cache storage system. Features of the COS design include:

- *Increased Disk Efficiency*. The read and write code paths have been optimized to increase the efficiency of disk drives participating in the cache. Similar reductions have been made to the create and delete paths. These optimizations are designed to allow the object store to use a single disk more efficiently and take full advantage of multiple drives on multiple channels.
- *Improved Spatial Locality*. The organization of content on disk has been optimized so that HTML and XML source documents and their embedded objects are likely to be located in the same region of the disk. This optimization makes it possible to provide automatic read-ahead capabilities to requesting user agents for composite pages.

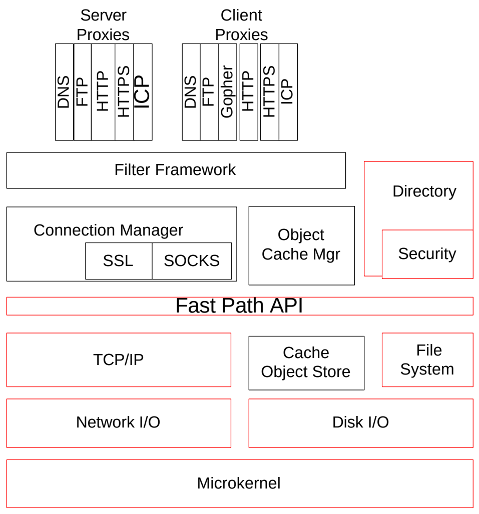

Figure 3. ICS Block Diagram.

Server Proxies communicate with the user agents. They act as servers, operating on behalf of web servers. Requests from user agents are processed and dispatched to the cache manager for processing. There is a server for each proxied application protocol. Currently there are server proxies for DNS Acceleration, FTP, FTP Acceleration, HTTP, HTTP Acceleration, HTTPS, and ICP.

Client Proxies communicate with web servers. Requests to web servers are made on behalf of the cache manager for cache misses, cache refreshes, and pass thru conditions. There is a client proxy for each application protocol. Currently there are client proxies for DNS, FTP, Gopher, HTTP, HTTPS and ICP.

A *Filter Framework* is provided for access control, filtration, and content distillation. It provides filter module extensions similar to those available in ISAPI [23] and NSAPI [29]. We chose to use the more efficient and scalable in-process model of execution used by ISAPI and NSAPI rather than the external process model used by CGI [26] and FastCGI [34]. Intrinsic filters are provided by the system for access control based upon PICS [41] rating labels. PICS ratings are loaded into the system from rules generated by the administrator and third-party PICS label providers. Access control granularity is provided for individual identity [15], group identity [15], DNS host name, IP address, and IP subnet.

4. MEASURED PERFORMANCE

Over the past two years, a number of performance and scalability measurements have been performed on ICS. Using established benchmarks, both web server acceleration and proxy cache performance measurements have been obtained. Live deployment of the system provides real world sample scale points.

4.1 Web Acceleration Benchmark

A study [31] measuring the peak performance of web acceleration (reverse proxy) using the WebBench 2.0 [42] standard workload found the ICS predecessor FastCache [30] capable of servicing 10,323 URL objects per seconds with a throughput of over 500Mbits/sec. An earlier benchmark submission to SPEC [39] demonstrated earlier versions of the system capable of very high performance. While these studies operate in controlled lab environments, they do ascertain peak acceleration performance capability. For instance, in our studies ICS was 5-10 times more scalable than other commercial systems.

Since a properly configured web accelerator serves the majority of its hit rate from RAM, these results isolate the network channel and cache from the disk and showcase the performance of ICS’s unified I/O buffering.

4.2 Proxy Cache Benchmark

In contrast to the study mentioned in the previous section, we performed a study [19] that measured steady state while using a working set far greater than would fit in RAM. This was done using the WCAT [24] benchmark tool and a very large workload derived from production logs, under a Zipf distribution data set size of 78GB.

Under these conditions ICS was capable of servicing 2,968 URL objects per second. ICS's ability to service these high rates of requests is directly due to COS, which allowed us to improve performance by 800% over the standard file system.

A recently completed study [37] done by the National Laboratory for Applied Network Research measured the steady state performance of several proxy caches under a constant offered request rate with a "Uniform" popularity model. Under these conditions, which did not utilize persistent connections, ICS was capable of servicing 1,500 requests per second with no degradation in response time.

4.3 Live Proxy Cache Deployment

The ICS predecessor FastCache has been in production service within the Utah education network (UtahLink) for the past two years. During this time, UtahLink has graciously participated in numerous pilot experiments that have produced empirical data for use in studies.

One of these studies [33] describes a single proxy cache supporting 87,000 workstations daily, with a user population of more than 400,000 and averaged 8,000 concurrent persistent connections during school hours.

5. FUTURE WORK

The research and development project described in this paper is part of an ongoing effort to produce large-scale Internet middleware applications and services hosted on Novell’s microkernel platform. The following are ongoing research activities.

We are exploring the suitability of the platform for other middleware services, including additional application proxies and brokers. More specifically, we are prototyping several data stream distillation applications which operate as inter-positioned filters running in conjunction with application proxies.

Fox, *et al*. [12] have advocated the use of Java in multi-tier web applications. We are exploring the suitability of Java middleware applications in conjunction with an advanced Java Virtual Machine.

The proxy cache service is capable of saturating several elements of industry standard hardware, such as the PCI bus, LAN controllers, disk controllers, and disk channels. To scale beyond a single system node, we have prototyped the use of cooperative service clustering and are evaluating it in a controlled deployment.

6. CONCLUSION

In this paper we have discussed ICS, a proxy cache service capable of serving tens of thousands of concurrent web users. This service is hosted on a specialized operating system designed to meet the rigorous demands of large-scale Internet middleware services. We have also described measured performance and scalability significantly beyond proxy cache services running on general-purpose operating systems.

7. ACKNOWLEDGEMENTS

The operating system platform and ICS are the result of many people’s efforts, too numerous to list here. Our thanks go to the anonymous WISP99 referees for their thoughtful comments on this paper, and to Bill Cox and Chip Webb who provided valuable comments and feedback on earlier drafts of this paper.

8. REFERENCES

[1] Almeida, J., and Cao, P. Measuring Proxy Performance with the Wisconsin Proxy Benchmark. *Proceedings of the 3rd International WWW Caching Workshop TF-Cache Meeting*. June 1998. Available from http://wwwcache.ja.net/events/workshop/27/cao-wpb.ps

[2] Almeida, J., Dabu, M., Manikutty, A.,and Cao, P. Providing Differentiated Quality-of-Service in Web Hosting Services. *Proceedings of the1st Workshop on Internet Server Performance*, June 1998. Available from http://www.cs.wisc.edu/~cao/WISP98/final-versions/jussara.ps.

[3] Anderson, E.W., and Pasquale, J. The Performance of the Container Shipping I/O System. *Proceedings of the Fifteenth ACM Symposium on Operating System Principles*, page 229, Copper Mountain, CO, Dec. 1995.

[4] Apache Organization. Apache HTTP Server Project. Available from http://www.apache.org/

[5] Banga, G., P. Druschel, P., and Mogul, J.C. Better operating system features for faster network servers. *Proceedings of the 1st Workshop on Internet Server Performance*, June 1998. Available from http://www.cs.wisc.edu/~cao/WISP98/final-versions/gaurav.ps.

[6] Cáceres, R., Douglis, F., Feldmann, A., Glass, G., and Rabinovich, M. Web Proxy Caching: The Devil is in the Details. *Proceedings of the 1st Workshop on Internet Server Performance*. June 1998. Madison, WI. Available from http://www.cs.wisc.edu/~cao/WISP98/final-versions/anja.ps

[7] Chankhundhod, A., Danzig, P.B., Neerdaels, C., Schwartz, M.F., and Worrell, K.J.. A Hierarchical Internet Object Cache. *Proceedings of the 1996 USENIX Technical Conference*. Jan. 1996.Available from http://www.usenix.org/publications/library/proceedings/ sd96/danzig.html

[8] Clark, D.D. The structuring of systems using up-calls. *Proceedings of the 10th ACM Symposium on Operating System Principles*, pages 171-180. Dec. 1985.

[9] Danzig, P. NetCache Architecture and Deployment. *Proceedings of the 3rd International WWW Caching Workshop TF-Cache Meeting*, June 1998. Available from http://wwwcache.ja.net/events/workshop/01/ NetCache-3_2.pdf.

[10] Duska, B.M., Marwood, D., and Feeley, M.J. The measured access characteristics of world wide-web client proxy caches. *Proceedings of USENIX Symposium on Internet Technology and Systems*, December 1997. Available from http://www.usenix.org/publications/ library/ proceedings/usits97/ duska.html

[11] DMTF organization. Directory Enabled Networks (DEN). Available from http://www.dmtf.org/spec/denh.html

[12] Fox, G.C., Furmanski, W., and Ozdemi, H.T. From Javabean to Database B Web Servers in the Pragmatic Web. *Proceedings of the 1st Workshop on Internet Server Performance*, June 1998. Available from http://www.npac.syr.edu/users/gcf/working/wiscpaper/ wisc98.ps

[13] Gien, M. Micro-kernel Architecture Key to Modern Operating System Design. Unix Review, 8(11):58-60. November 1990. Available from ftp://ftp.chorus.fr/pub/reports/CS-TR-90-42us.ps.Z

[14] Hamilton, G., and Kougiouris, P. The Spring nucleus: A microkernel for objects. *Proceedings of the 1993 Summer Usenix Conference*, pages 147-159, Cincinatti, OH, June 1993.

[15] Housley, R., Ford, W., Polk, W., and Solo, D. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. IETF RFC2459. Available from ftp://ftp.isi.edu/in-notes/rfc2459.txt.

[16] IETF organization. Policy Framework (policy) working group. Available from http://www.ietf.org/html.charters/policy-charter.html

[17] Jones, M.B., Leach, P.J., Draves, R.P., and Barrera, J.S. Modular real-time resource management in the Rialto operating system. *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems)HotOS-V)*, Orcas Island, WA. May 1995.

[18] Kaashock, M.F., Engler, D.R., Ganger, G.R., Briceno, H., Hunt, R., Mazieres, D., Pinckney, T., Grimm, R., Janotti, J., and Mackenzie, K. Application performance and flexibility on Exokernel systems. *Proceedings of the Sixteenth Symposium on Operating System Principles*, Saint-Malo, France. Oct. 1997.

[19] Lee, R., and Tomlinson, G. Workload Requirements for a Very High-Capacity Proxy Cache Design. *Work-In-Progress of the 4th International Web Caching Workshop*, March 1999. Available from http://www.ircache.net/Cache/Workshop99/Papers/lee-0.ps.gz.

[20] Major, D., Minshall, G., and Powell, K. An Overview of the NetWare Operating System. *Proceedings of USENIX Winter 1994 Technical Conference*., January 1994. Available from http://www.usenix.org/publications/library/proceedings/sf94/minshall.html.

[21] Major, D., et al. Specialized Operating System Design: The Novell Microkernel. In Preparation.

[22] Major, D., et al. The direct unified I/O framework employed by NetWare. In Preparation.

[23] Microsoft Corporation. Internet Server API (ISAPI) Reference. Available from http://www.microsoft.com/win32dev/apiext/isapiref.htm

[24] Microsoft Corporation. Web Capacity Analysis Tool (WCAT). Available from http://msdn.microsoft.com/subscriptions/index/jan99/Universal/BackOffice/Development/U.S/1443.asp.

[25] Mogul, J.C. Operating system support for busy internet servers. *Proceeding of the Fifth Workshop on Hot Topics in Operating Systems (HotOS-V)*, Orcas Island, WA. May 1995.

[26] National Center for Super Computing Applications(NCSA), University of Illinois. Common Gateway Interface. Available from http://hoohoo.ncsa.uiuc.edu/cgi/

[27] National Center for Super Computing Applications, University of Illinois. NCSA HTTPd server. Available from http://hoohoo.ncsa.uiuc.edu/

[28] National Laboratory for Applied Network Research. Squid Internet Object Cache. Available from http://squid.nlanr.net/

[29] Netscape, Inc. Netscape API Function Reference. Available from http://developer1.netscape.com:80/docs/manuals/enterprise/nsapi/deffunc.htm

[30] Novell, Inc. BorderManager FastCache Services 3, Internet Acceleration Software. Available from http://www.novell.com/bordermanager/fastcache/

[31] Novell Inc. Novell Demos 10,000 Hits-per-Second Web Server Acceleration. Novell Research Note, October 1998. Available from http://www.novell.com/bordermanager/ispcon4.pdf.

[32] Novell, Inc. Internet Caching System. Available from http://www.novell.com/products/nics/

[33] Novell, Inc. Single Proxy Server Supports 67,000 seat Network for Utah Schools. Novell Research Note, December 1997. Available from: http://www.novell.com/bordermanager/utahlink.html

[34] Open Market, Inc. Fast CGI. Available from http://www.fastcgi.com/

[35] Pai, V.S., Druschel, P., and Zwaenepoel, W. IO-Lite: A unified I/O buffering and caching system. Technical Report TR97-294, Rice University, CS Dept., Houston, TX, 1997. Available from http://www.cs.rice.edu/~vivek/IO-Lite/TR97-294.ps

[36] Ritchie, D.M., and Thompson, K. The UNIX Time-sharing System. Bell System Technical Journal, 57(6):1905-1929.

[37] Rousskov, A., Wessels, D., and Chisholm, G. The First IRCache Web Cache Bake-off, The Official Report. *Proceedings of the 4th International Web Caching Workshop*, March 1999. Available from http://bakeoff.ircache.net/bakeoff-01/report.ps.gz

[38] Thadani, M.N., and Khalidi, Y.A.. An efficient zero-copy I/O framework for UNIX. Technical Report SMLI TR-95-39, Sun Microsystems Laboratories, Inc., May 1995. Available from http://www.sunlabs.com/technical-reports/1995/ smli_tr-95-39.ps

[39] The Standard Performance Evaluation Corporation. SPECWeb 96 Results - "HP Hits 2131 SPECweb96 ops/sec. with BorderManager Web Server Acceleration on a 400MHz NetServer LH - May 98". Available from http://www.spec.org/osg/web96/results/res98q2/web96-980518-02768.html.

[40] Wahl, M., Howes, T., and Killie, S. Lightweight Directory Access Protocol (v3). IETF RFC2251. Available from ftp://ftp.isi.edu/in-notes/rfc2251.txt

[41] World Wide Web Consortium. PICS Label Distribution Label Syntax and Communication Protocols Version 1.1. REC-PICS-labels-961031, October 1996. Available from http://www.w3.org/TR/REC-PICS-labels.

[42] Ziff Davis Inc. WebBench 2.0. Available from http://www.zdnet.com/zdbop/webbench/webbench.html