

# Performance Issues in WWW Servers

## (Extended Abstract)

Erich Nahum, Tsipora Barzilai, and Dilip Kandlur  
IBM T.J. Watson Research Center  
Hawthorne, NY 10532  
{nahum,tsipora,kandlur}@watson.ibm.com

### Abstract

This paper evaluates performance issues in WWW servers on UNIX-style platforms. While other work has focused on reducing the use of kernel primitives, we consider ways in which the operating system and the network protocol stack can improve support for high-performance WWW servers. We study techniques in 3 categories: new socket functions, per-byte optimizations, and per-connection optimizations. We examine two proposed socket functions, `acceptex()` and `send_file()`, comparing `send_file()`'s effectiveness with an `mmap()/writev()` combination. We show how `send_file()` provides the necessary semantic support to eliminate copies and checksums in the kernel, and quantify the utility of the function's header and close options. We also present mechanisms to reduce the number of packets exchanged in an HTTP transaction, both increasing server performance and reducing network utilization, without compromising interoperability. We evaluate these issues with a high-performance WWW server, using IBM AIX workstations connected over 100 mbps Ethernet, driven by the WebStone and SURGE WWW server workload generators. Microbenchmark results using WebStone show that our combination of mechanisms can improve server throughput by up to 53 percent, and can eliminate up to 33 percent of the packets in an HTTP exchange. Macrobenchmark results with SURGE show an aggregate increase in server throughput of 25 percent.

### 1 Introduction

The phenomenal growth of the World-Wide Web, in both the volume of information on it and the numbers of users desiring access to it, is dramatically increasing the performance requirements for large scale information servers. WWW server performance is thus a central issue in providing ubiquitous, reliable, and efficient information access. This paper evaluates issues in WWW server performance on UNIX-style platforms. While other work has focused on reducing the use of kernel primitives, we explore ways in which the operating system and the network protocol stack can improve support for high-performance WWW servers. Issues we consider include:

- *new socket functions.* Microsoft has added two new socket functions to NT [4], `acceptex()` and `transmitfile()`, and HP has a similar function `send_file()` in HP-UX. These APIs streamline the programming interface used by a web server in a typical HTTP transaction, but do they provide any performance benefit? Does `transmitfile()` or `send_file()` show any improvement over the already available `mmap()` and `writev()` system calls?
- *per-byte optimizations.* It is well-known that data touching operations, such as copying and checksumming, are expensive. BSD-derived Unix operating systems use different buffering mechanisms in the file system and the networking code, forcing data to be copied when it is moved from one subsystem to another. How well can we approach a zero-copy integrated I/O architecture [7], while continuing to exploit the benefits of existing file systems? What sort of performance impact will eliminating data touching operations have for WWW servers?
- *per-connection optimizations.* TCP connection management was not designed for client-server traffic, exchanging more packets than is semantically necessary. While the transition to persistent connections in HTTP 1.1 will improve this, most machines are still using HTTP 1.0. How can the per-connection overhead be reduced, without violating the TCP protocol specification?

We study these issues using a testbed of several IBM RS/6000 AIX workstations connected over 100 mbps Ethernet. We use Rice University's Flash WWW server, which exploits most currently-known user-level optimizations, and utilize the WWW workload generators WebStone and SURGE [3] to drive the system with HTTP 1.0 requests.

Our experience confirms previous work showing that WWW servers spend most of their time in the kernel [1, 5]. We build upon previous work by studying ways in which the operating system and protocol stack can improve support for high-performance WWW servers. We examine the benefits of two proposed socket functions, `acceptex()` and `send_file()`, comparing the effectiveness of `send_file()` with an combination of `mmap()` and `writev()`. We show how `send_file()` provides the necessary semantic support for further optimizations, such as eliminating copies and reducing packet exchanges, and quantify the utility of the function's header and close options. We present mechanisms to reduce the number of packets exchanged in an HTTP transaction, both increasing server performance and reducing network utilization.

| File Size<br>(bytes) | Flash<br>Base | Flash<br>Optimized | Diff<br>(%) |
|----------------------|---------------|--------------------|-------------|
| 1024                 | 1140.11       | 1407.55            | 23.46       |
| 2048                 | 1059.26       | 1320.16            | 24.63       |
| 4096                 | 904.15        | 1166.91            | 29.06       |
| 8192                 | 722.22        | 937.51             | 29.81       |
| 16384                | 501.92        | 708.30             | 41.12       |
| 65536                | 187.72        | 298.16             | 58.83       |
| 262144               | 54.36         | 89.15              | 64.00       |
| 1048576              | 13.55         | 21.79              | 60.81       |

Table 1: HTTP Throughput in ops/sec (WebStone)

| Configuration              | SURGE<br>Ops/sec | Diff<br>(%) |
|----------------------------|------------------|-------------|
| Flash-Poll                 | 437.72           |             |
| + <code>send_file()</code> | 418.05           | -05         |
| + Mbuf Caching             | 519.83           | +20         |
| + Checksum Offload         | 555.14           | +06         |
| + FIN Piggyback            | 560.66           | +01         |
| + Delayed Ack of FIN       | 571.60           | +02         |
| + Delayed Ack of SYN       | 581.56           | +02         |
| Total Improvement:         |                  | +25         |

Table 2: HTTP Throughput in ops/sec (SURGE)

## 2 Overview of Results

Space constraints prevent us from describing our results fully, thus we can only provide an overview of our findings. Table 1 shows how our optimizations improve server throughput across requests for different file sizes as measured by WebStone. Table 2 shows the aggregate increase in server throughput as our optimizations are incrementally added as measured by SURGE. Interested readers should consult the IBM research report [6] for more details. We summarize our findings as follows:

- *new socket functions.* We evaluate the proposed socket functions `acceptex()` and `send_file()`. We find little or no increase in performance using the `acceptex()` function, on either process-based or thread-based WWW servers. In addition, kernel profiling shows that servers spend a relatively small amount of time in the `accept()`, `getsockname()`, and `read()` system calls. A `send_file()` implementation that incurs a single copy provides no advantage over a combination of `mmap()` and `writew()`.
- *per-byte optimizations.* Per-byte optimizations that we examine include eliminating a data copy on the fast path by caching mbuf's within the kernel and offloading the TCP checksum to the adaptor. A `send_file()` implementation tied to an integrated I/O system which does not copy data provides substantially better performance. In our testbed, we observe an increase in throughput of up to 53 percent. We find that offloading the checksum to the network device can improve WWW server performance by up to 7 percent. Our mbuf cache mechanism can also be enhanced to allow caching of the checksum values in the mbufs, for network interfaces that do not support the checksum offload.
- *per-connection optimizations.* Our per-connection optimizations reduce overhead by eliminating redundant packets in the TCP connection setup and teardown. We show how the close option to `send_file()` provides the semantic support to enable piggybacking the FIN on the last data segment, eliminating one packet in small transfers and improving throughput by 6 percent in those cases. We also show how delaying acknowledgments for the FIN and SYN-ACK packets can eliminate 2 more packets, increasing performance an additional 14 percent for small transfers. In total, we reduce the packets in a small HTTP exchange from 9 to 6, reducing network utilization and raising server throughput by up to 20 percent in those scenarios, all without violating the TCP protocol specification. Our changes are more easily deployed

incrementally since they do not require both hosts in a conversation to adopt them, unlike T/TCP or SACK.

- *aggregate benefits.* Using SURGE as a macrobenchmark, we show that the combination of techniques improve aggregate server performance by 25 percent.

While we have evaluated these optimizations in the context of a WWW server, they have utility for other programs as well. Reducing packet exchanges should help other TCP-based applications, and `send_file()` is a general function that can be used by other network servers, such as NFS, FTP, or SMB. As a consequence of our findings, IBM's AIX division has released these features in AIX 4.3.2.

## References

- [1] Jussara M. Almeida, Virgilio Almeida, and David J. Yates. Measuring the behavior of a world-wide web server. In *Seventh IFIP Conference on High Performance Networking (HPN)*, White Plains, NY, April 1997.
- [2] Martin F. Arlitt and Carey L. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5):631–646, Oct 1997.
- [3] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Madison, WI, June 1998.
- [4] James C. Hu, Irfan Pyarali, and Douglas C. Schmidt. Measuring the impact of event dispatching and concurrency models on web server performance over high-speed networks. In *Proceedings of the 2nd Global Internet Conference (held as part of GLOBECOM '97)*, Phoenix, AZ, Nov 1997.
- [5] Yiming Hu, Ashwini Nanda, and Qing Yang. Measurement, analysis, and performance improvement of the Apache web server. Technical Report 1097-0001, University of Rhode Island Department of Electrical and Computer Engineering, Oct 1997.
- [6] Erich M. Nahum, Tsipora Barzilai, and Dilip Kandlur. Performance issues in WWW servers. IBM Research Report, May 1999.
- [7] Vivek S. Pai, Peter Druschel, and Willy Zwaenepoel. I/O Lite: A copy-free UNIX I/O system. In *3rd USENIX Symposium on Operating Systems Design and Implementation*, New Orleans, LA, February 1999.