Performance Anomalies of Advanced Web Server Architectures in Realistic Environments

Takashi Shinozaki  Eiji Kawai  Suguru Yamaguchi  Heiichi Yamamoto
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 6300101 JAPAN
Email: {takash-s, eiji-ka, suguru, heiichi}@is.naist.jp

Abstract — When we discuss performance of network servers, simple benchmark tests can mislead us into inaccurate and wrong results. In this paper, we present precise results of web server performance evaluations on emulated networks whose parameters include network delay and packet loss ratio. We chose three types of target web servers: Apache, thttpd, and TUX. Their software architectures are largely different from each other and their pros and cons are discussed from a general viewpoint in a number of previous literatures. However, we still found some performance anomalies in the experimental results, which revealed implementation issues in those servers.

Keywords — Performance, web, server, network, delay, loss, Apache, thttpd, TUX.

1. Introduction

The growth of network bandwidth is much larger than that of computer performance. Therefore, the scale of a server cluster system deployed by popular web sites also grows year by year, and this makes the management cost of such a large scale web server cluster explosive.

In this situation, not only technologies that make a server cluster system scalable but also technologies that increase the performance of each server system are absolutely essential. Especially, many research groups are developing new server architectures and new operating system features to achieve higher performance. The technologies developed in those groups include zero-copy file transfers (the sendfile() system call), interrupt coalescing, a variety of offloading mechanisms (large segment offloading: LSO, checksum offloading, and TCP offloading engine: TOE), and advanced server architectures (multi-process/multi-thread: MP/MT, single process event-driven: SPED, and in-kernel server implementations) [1, 2, 3, 4].

Although the technologies listed above was proved to achieve high performance by benchmark tests in their research experiments, the experiments often lacked a viewpoint of network delays and packet losses, which hides the performance anomalies in real-world server implementations. In our research, we conducted benchmark tests in realistic network environments. We utilized three typical web server implementations with advanced architectures: Apache web server [5], thttpd [6], and TUX [7]. Our approach reveals real problems in state-of-the-art server implementations. The final goal of this research is to resolve these problems and make the servers more scalable and the results obtained in this paper is the first step to that goal.

2. Web Server Implementations

As we already mentioned, the Apache, thttpd, and TUX web servers were chosen as target for the performance evaluation. In this section, we briefly describe their software architectures.

2.1. Apache

The Apache web server is based on a multi-process (MP) architecture. Since each client connection is associated with a different process, this architecture spawns a large number of processes to support a large number of concurrent clients, which makes the process scheduling overhead considerably high. Therefore, the performance scalability of this architecture is lower than that of the others.

2.2. thttpd

The thttpd web server is based on a single process event-driven (SPED) architecture. This architecture requires only a small number of processes, possibly one process, to support a large number of concurrent clients by utilizing the polling I/O mechanism, which is usually implemented by select() and poll() in Unix variants. Therefore, the performance scalability of this architecture is much higher than that of the MP architecture.

2.3. TUX

The TUX web server (Red Hat Content Accelerator) is a kernel-based high performance server developed by RedHat. Because this architecture removes various overheads caused by system calls such as processor privilege switching, arguments checking, and data copies between the user-land and the kernel-land. In addition, it implements various mechanisms to avoid data copying. For example, it removes data copies between the file system and the protocol stack by sending data directly from the page cache in the file system. It also removes data copies in the protocol stack by bypassing the socket abstraction layer for network writes.
3. Benchmark Tests

We conducted benchmark tests on web servers in emulated network environments. Figure 1 shows the experimental system environments including a web server, a network router that emulates network delays and packet losses, and clients that put a wide-variety of load on the server. The hardware spec of these hosts are listed in Table 1. In this section, we describe the configurations of the experiments.

3.1. Network Environments

We utilized the NIST Net network emulator [8] in the router that connects the clients and the server. In our experiments, we configured it to put network delays of 0 ms, 15 ms, and 50 ms for each way between the clients and the server (therefore, the RTTs are 0 ms, 30 ms, and 100 ms, respectively), and packet loss ratios of 0%, 1%, and 4%.

3.2. Benchmark Test Parameters

We utilized the httperf web benchmark tool [9] to put a wide variety of loads on the server. We prepared content files whose sizes were 16 KB and 1 MB. With the httperf software, we can obtain various performance indices including reply rates and reply times.

3.3. Kernel Profiling

In our experiments, we also conducted kernel profiling on the server host with OProfile [10], a system-wide profiler for Linux systems. The largest benefit we can receive from the OProfile is the precise information of the execution cost, especially in the software/hardware interrupt handlers. Because a large part of the TUX web server is executed in the interrupt handler context to achieve low service delay, this feature is highly important for this study.

4. Results

We first show the overview of the experimental results, and then discuss two important performance anomalies found in the results.

4.1. Overview of Results

First, we present experimental results when the data size was 16 KB. Figure 2 and 3 depict reply rates in replies per second with RTT of 0 ms and 100 ms, respectively. The results when the RTT was configured at 30 ms are omitted because of the limited space. Figure 4 and 5 show mean reply times in milliseconds.

These results give supplementary proof that the performance of TUX is the highest and that of Apache is the lowest among

![Figure 2. Reply Rate: data size = 16 KB, RTT = 0 ms, loss ratio = 0% (left), 1% (center), 4% (right)](image)

![Figure 3. Reply Rate: data size = 16 KB, RTT = 100 ms, loss ratio = 0% (left), 1% (center), 4% (right)](image)
these three web servers, which we briefly described in Section 2. On the other hand, we can observe performance degradation of Apache and thttpd with a large packet loss ratio. We will see this phenomenon in detail later.

Next, we see the results when the data size is 1 MB. Similar to the previously shown graphs, Figure 6 and 7 are for reply rates, Figure 8 and 9 are for reply times.

Compared to the results when the data size is set at 16KB, we cannot see obvious difference in reply rates of three web servers. This is because the performance bottleneck moved from server implementation (i.e., efficient processor utilization) to available network bandwidth. Consider a case when a server transmits an 1 MB file for 70 times a second, the total bandwidth exceeds 500 Mbps. In this case, the number of sessions the server has to manage during the test is small and this also means that the efficiency of server architecture has lower importance. Thus, the server performance is limited by other factors than the server implementation such as PCI bandwidth,
TCP/IP protocol stack, interrupt handling, etc.

On the other hand, when we see the reply times in Figure 8 and 9, we find ridiculous performance anomalies; Apache returned responses fastest and TUX did slowest. We will see this phenomenon in the next subsection.

4.2. Performance Anomalies

As we already mentioned, there are two performance anomalies found in this study. One is the large performance degradation of user-land web servers when a small file (16 KB) was transferred through a link with a large packet loss ratio (4%).

Especially, thttpd slowed down over 70%. The reason of this phenomenon is the increased session number the server had to handle concurrently. Both Apache and thttpd have poor scalability against the number of the concurrent sessions because of their software architecture reasons. Apache allocates a single process to each session and thus the number of sessions are limited by the number processes that can be spawned and managed in the system where it runs. Although thttpd can handle a large number of concurrent sessions more efficiently than Apache, it utilizes the polling I/O mechanism and the overhead in scanning the descriptor table grows larger when the number of concurrent sessions increases. Actually, we examined the results obtained from OProfile and found that about 40% of CPU cycles were consumed by polling I/Os in the operating system kernel.

One of the solutions to this problem is to adopt an explicit event delivery mechanism. For example, real-time signals implemented in many Unix variants today are a good candidate for the solution. In addition, many operating systems implement their own mechanisms such as the kqueue mechanism in FreeBSD, the epoll mechanism in Linux, and the polling device mechanism in Solaris.

The other anomalies we found in this study is the large response time of TUX when a large file (1 MB) was transferred. Figure 10 and 11 represent the execution profile obtained from OProfile in the tests. We categorized in-kernel functions into buffer management, copy and checksum calculation, interrupt handling, system call processing, TCP/IP processing, device drivers, and the others. Although we collected profile data throughout the experiments, OProfile did not work well with Apache under overloaded situations. Therefore we put only valid data in those graphs.

From the graphs, we cannot find any major difference especially between the thttpd and TUX behaviors. This implicates that the large response time of TUX is caused by its session management implementation in itself and not by the kernel implementation.

To investigate this phenomenon more precisely, we added a few small modifications into the httperf program, which are listed bellow, to obtain time series data, and conducted extra experiments with a lengthened test duration time (12 minutes).

- Output reply rate in each second.
- Discard the initial and last several minutes sample data to increase the accuracy of the results.

Figure 12 depicts precise histogram of the reply times. As shown in these graphs, user-land servers achieved short response time in many responses. On the other hand, all the response times of TUX exceeded 26 seconds. Table 2 shows some basic statistics of the connection times. Those connection times involve the data from timeout sessions which were excluded from the reply time depicted in Figure 12.

Figure 13 represents the reply rates obtained from httperf each second. From the graph, the TUX shows considerably
different behavior from that of the other servers. Many sessions in TUX were completed in specific seconds.

Table 2. Connection Time (ms)

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Max</th>
<th>Average</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>1.782</td>
<td>104.652</td>
<td>18.303</td>
<td>19.707</td>
</tr>
<tr>
<td>thttpd</td>
<td>10.168</td>
<td>80.460</td>
<td>25.807</td>
<td>13.807</td>
</tr>
<tr>
<td>TUX</td>
<td>26.492</td>
<td>105.840</td>
<td>49.210</td>
<td>15.691</td>
</tr>
</tbody>
</table>

Considering these graphs, we can conclude that the fairness in the session management implemented by TUX is better than that of the other servers. However, we can also conclude that the internal synchronization of network I/O processing is observed in TUX. To study more precise behavior of TUX, other research schemes will be required such as packet monitoring.
5. Open Issues

In this paper, we presented two performance anomalies found in our experiments. However, there remains some open issues for future high-performance server development.

5.1. More Sophisticated Benchmark Tools

The httperf software is one of the most powerful benchmark tools for web server performance evaluation. However, more detailed examination of server behavior is required to investigate real-world server architecture development.

Especially, it is an extremely difficult issue to evaluate the performance of an overloaded server. We can configure the server system to be able to establish a large number of concurrent sessions. However, the service throughput of the system is limited and therefore the client systems have to maintain a large number of responseless sessions. All the server can do is to process the sessions in a best effort manner even when it is overloaded. On the other hand, the clients should provide a kind of guarantee that it is accurate enough even when the server is overloaded and most sessions between the client and the server are idle. Most web server benchmark tools can estimate the maximum service throughput. However, there are few ones that can evaluate the performance of overloaded servers.

5.2. Fairness of Session Management in Large Scale Servers

Because of rapid growth of network bandwidth, the number of sessions a server system handles concurrently increases largely. In many literatures, fairness of flow control mechanisms is discussed deeply such as various TCP flow control algorithms. On the other hand, the fairness issue is rarely considered as server implementation issue. Especially, we should be conscious that a future server system will handle tens of thousands or more of long-lived concurrent sessions on a single host. We should establish a good session management model in such a scenario.

6. Summary

In this study, we have conducted benchmark experiments to evaluate the performance of the Apache, thttpd, and TUX web servers under various factors such as file sizes, network delays, and packet loss ratios. In the experimental results, we found some performance anomalies. Especially, the reply times of the TUX web server grew much longer than those of the others when the request rate exceeded its maximum reply rate. To investigate the causes of the longer reply time, deeper analyses are required and they are our main future work.

REFERENCES


