

Frequency Domain Analysis and Visualization of Web Server Performance

Marc Chiarini and Alva Couch

Tufts University

BACKGROUND

After many years of study, performance tuning of web servers remains more of an art than a science. One is often reduced to guessing what will improve performance via trial and error. Compounding this difficulty, we find that simple textbook models of web server behavior do not adequately describe the relationships between caching, resources, and performance. In this study, we looked closely at web server performance with the goal of defining and understanding normal behavior; we took measurements of a minimally configured web server expecting to see textbook curves of response time. Instead, we observed complex curves that demonstrate in theory that transactions are not satisfied in a "memoryless" manner, and in practice seem to be related to caching behavior. We believe that even a cursory study of these curves may lead to better models of server normalcy and may allow the everyday sysadmin to more quickly analyze web server failures and performance deficiencies.

Part of the reason that our results differ so much from classical results is that we utilized a novel measurement strategy. Rather than relying upon server logs, we employed the Linux auditing subsystem, which is normally intended for security logging. The auditing system allowed us to log the time taken to satisfy each request, rather than just the time at which the request entered the system or arrived at the client. We were then able to visualize the frequency of response times to a request. This strategy revealed that even simple web servers whose behavior should be straightforward exhibit unforeseen and sometimes unexplained complexity of response.

METHODOLOGY

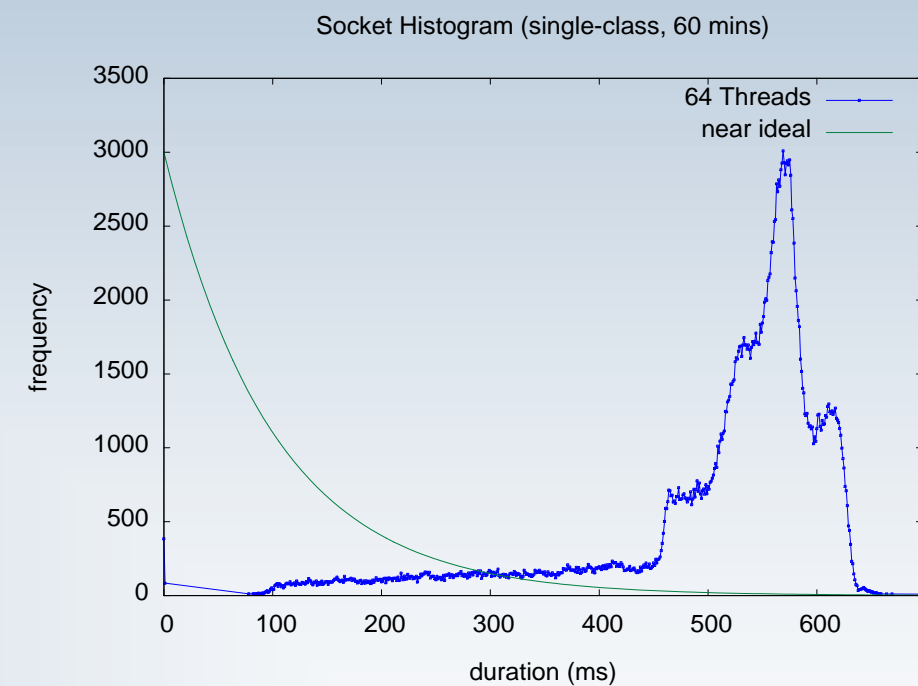
We took several steps to build an observational (external) model of web server behavior:

First, we constructed an environment with a single RHEL 4 Linux server running Apache 2.0.52 on a modest 2.4Ghz PC with 1G of RAM. Clients ran Jmeter (a Java desktop application designed to load test functional behavior and measure performance) on more powerful multi-core machines to execute a series of test plans in various configurations. An initial set of plans tested the response to a single class of HTTP requests (i.e. a single file, or multiple files of the same length) under various loads (concurrent threads sending requests without delay). Additional plans tested multi-class behavior where requests were made for files of widely varying sizes.

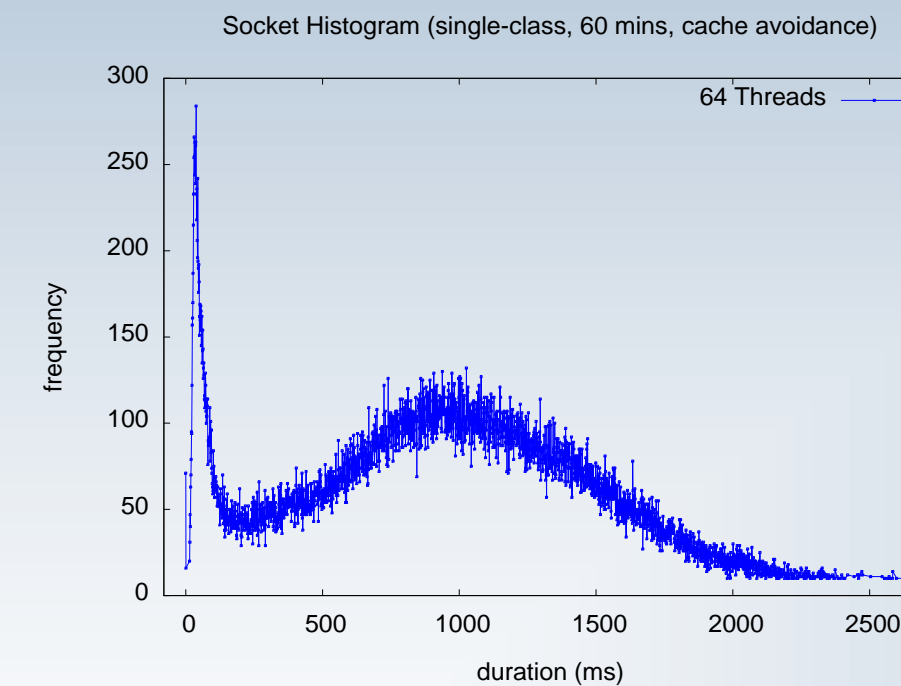
Next, we designed software in C and PERL to collect and post-process statistics about incoming HTTP requests generated by JMeter, as well as how the server responds in various dimensions, such as load average, CPU, memory, network, and disk utilization.

Although the web server was configured not to cache file requests, the operating system and the disk subsystem maintain their own caches. We wondered how much these cache sizes affect server performance, and hypothesized that the dominant effect causing the response curves seen here was caching. It was not technically feasible for us to disable the kernel and disk caches, so we tested our hypothesis instead via a "cache avoidance" technique. This consisted of reproducing the requested files in numbers that were sufficient to force the OS to retrieve them from disk on every request. Cache avoidance resulted in very different curves that were closer to theoretical performance models, so we conclude that kernel caching was a dominant factor in the common case. It remains to be seen how much the audit subsystem and statistics collector, which both dirty these caches, might affect this scheme.

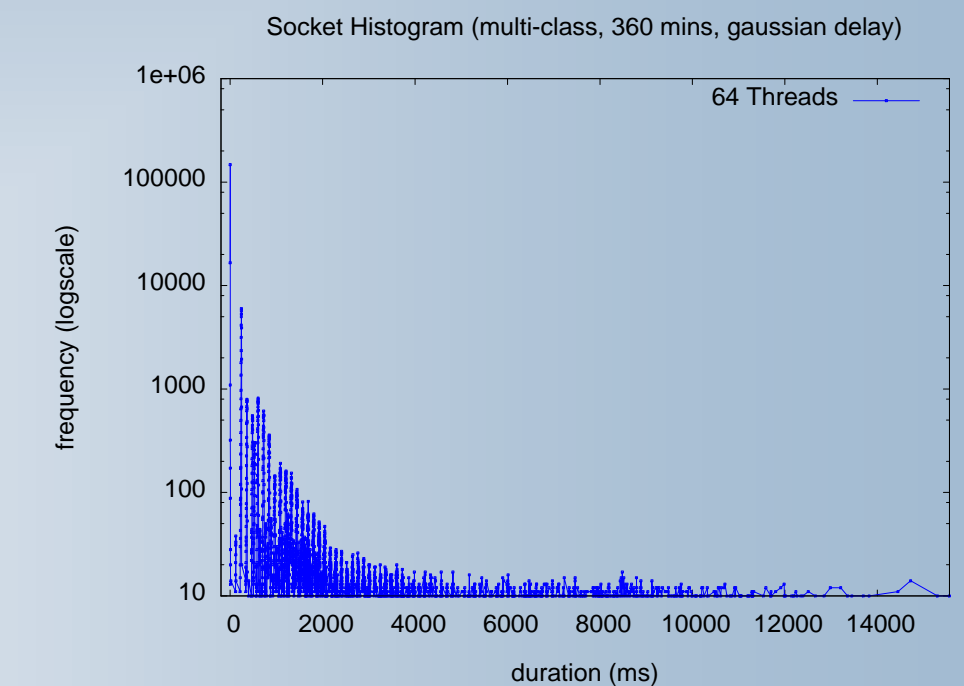
RESULTS



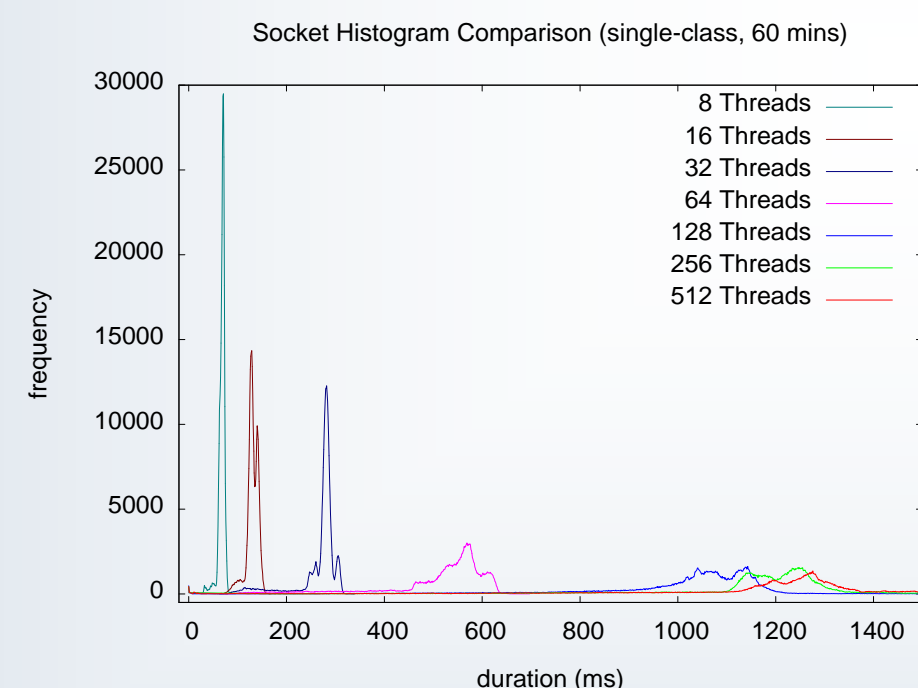
A histogram of socket durations over 60 minutes with 64 threads asking for the same file over and over. A majority of requests are completed between 460 ms and 630 ms, with the median service time around 547 ms. The green curve represents nearly ideal exponential service. We believe caching is the main reason these curves differ so greatly.



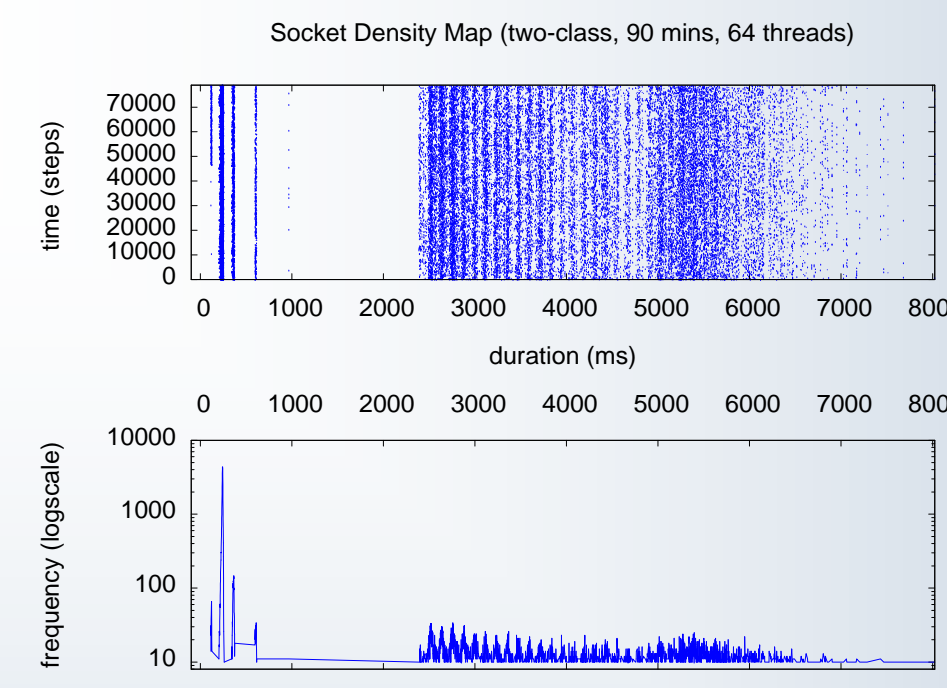
Using cache-avoidance gets us closer to the ideal. Most of the differences in this case seem to come from rotational latency of the disk as it quickly serves one file after the other. We actually get a more even service time distribution, subject to the layout of the files on disk and the disk drive's capabilities. The median service time is 956 ms.



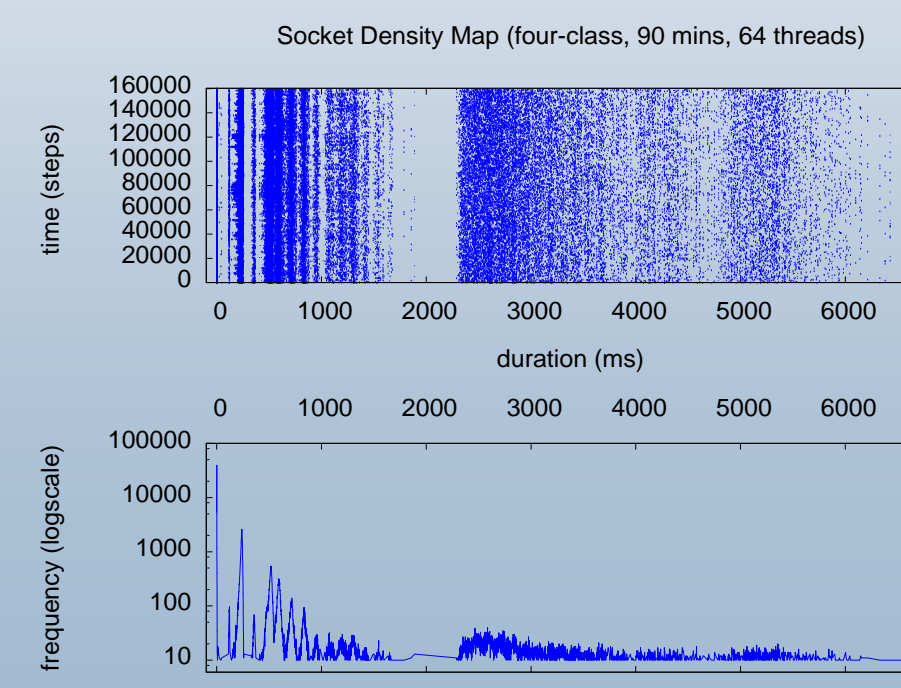
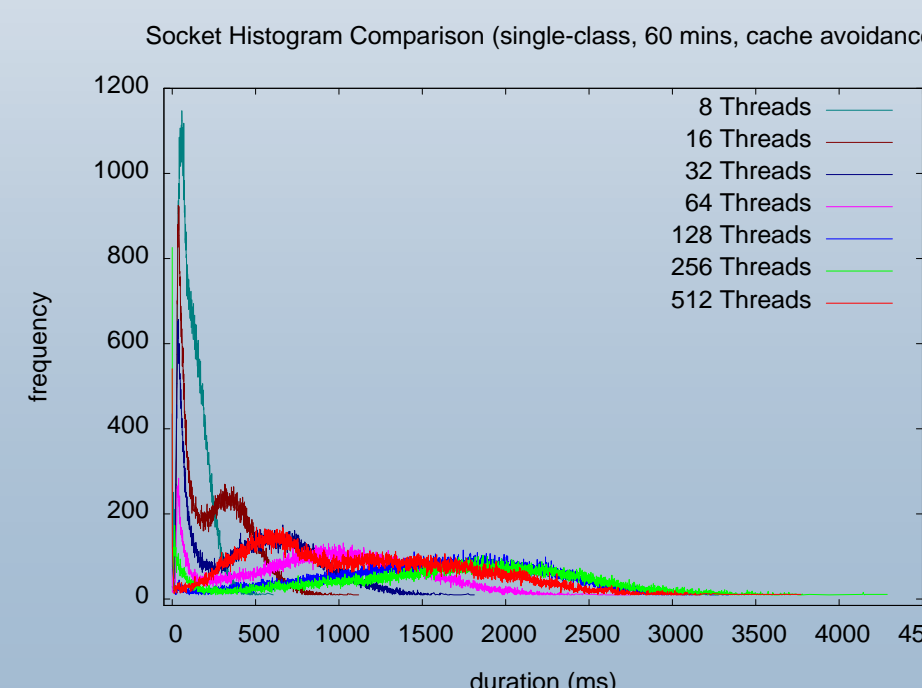
When we move to tests of longer duration and introduce Gaussian delay (up to 100 ms) between multiple classes of requests, our response curve starts to look much better.



A comparison of socket duration histograms for single-class requests under increasing loads. The mean response time doubles after each doubling of the load until 128 threads. This tells us that beyond a 64 thread load, the server is reaching a saturation point. Although less visible below, saturation also occurs using the cache avoidance technique.



Socket density maps are another useful way to visualize web server performance. This type of graph shows one point for every actual socket call. At a glance, one can view how long the classes of requests are taking (dark bands) and verify one's assumptions with the histogram.



CHALLENGES

There remain several challenges to utilizing our techniques in the field:

- Sysadmins will need to learn how to interpret what they are seeing and how the curves are affected in various circumstances.
- We must build a taxonomy of common behaviors to be used as a starting point in troubleshooting performance problems and optimizing existing configurations.
- It would be highly desirable to integrate frequency domain analysis into a larger monitoring infrastructure such as Nagios.
- We must assess the performance impact of the data collection infrastructure and determine if it introduces any deleterious phenomena (such as the disk latency mentioned in the graphs).

CONCLUSIONS

This research has verified that even simple web service exhibits complex performance behavior, far removed from theoretical models. The most important results for the system administration community come from the novel method we have introduced for visualizing performance. We believe that a frequency domain analysis infrastructure can be built around our initial research that would aid site administrators to both tune and troubleshoot their servers.