zeus

White Paper

Accelerating Apache with Zeus Traffic Manager

Improving Apache Performance

The Apache Web Server is the Internet's most popular web server software, as a result of its maturity, free distribution and the enormous range of community-supported modules. It is deeply embedded into many organisations' internet hosting infrastructures. However, Apache has several performance limitations that can result in poor and uneven end-user service, inefficient resource utilisation and higher hardware and administration costs.

This report investigates the performance limitations in the Apache Web Server and describes, with detailed benchmarks, how the Zeus Traffic Manager software can overcome these deficiencies. Key findings include:

- Apache's Keepalive Implementation gives very inconsistent levels of service when under load. Zeus Traffic Manager can manage Keepalives on Apache's behalf to give **even and consistent levels of service**.
- Apache's performance when not using Keepalives is poor, with large error rates and low transaction rates. Using Zeus Traffic Manager with or without Keepalives totally eliminates errors, and results in an increase of up to **18-times the sustained transaction rate**.
- Apache's SSL performance is sub optimal, with slow transaction times, limited capacity and connection errors under load. Using Zeus Traffic Manager to decrypt SSL traffic provides up to **20-times the transaction rate** and **20-times faster transactions**, with no connection errors.
- Apache performs very poorly on real-world high-latency networks. Zeus Traffic Manager almost totally eliminates the high-latency effects, giving up to **40-times better utilisation**, and **8-times faster transaction times**.

Test Background

Broadband-Testing conducted a series of tests to investigate the performance profile of the Apache server, and the effect of using Zeus Traffic Manager to accelerate transactions on the server.

The tests evaluated Apache 2.0.56 running on RedHat Enterprise Linux release 4.0 on Sunfire v20Z servers with 2Gb memory and Opteron 244 processors.

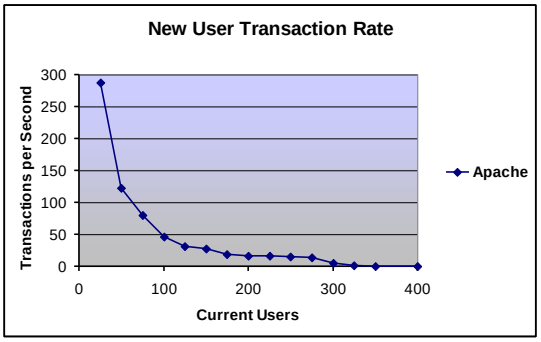
Tests conducted by Broadband-Testing used Spirent Avalanche test equipment and a single processor v20Z server. Other tests in this report were conducted by Zeus Technology, using apachebench or zeusbench (an apachebench equivalent), with a dual-processor v20Z server.

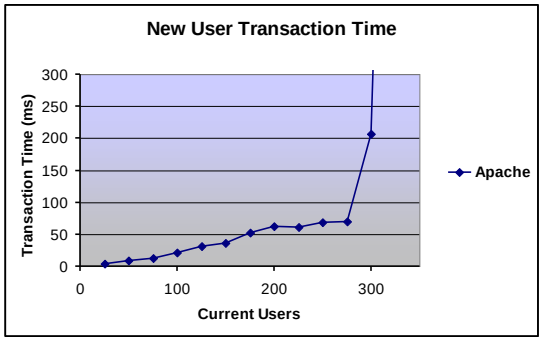
The Zeus Traffic Manager system was a 'ZXTM 7000 appliance' – an AMD Opteron system running a standard Linux kernel.

Zeus gratefully acknowledges the assistance and expertise of the Broadband-Testing team who oversaw and validated many of the tests in this report.

Key Finding 1: Uneven levels of Service under load

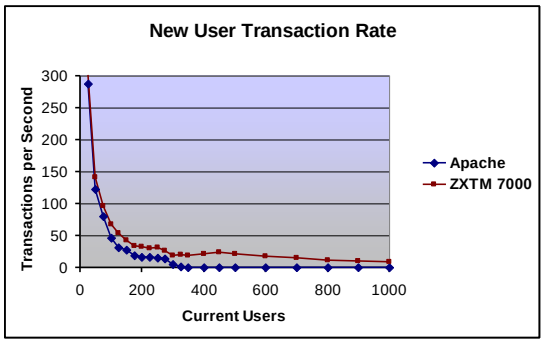
The Apache server gives very uneven levels of service under load. The following test used apachebench to load up an Apache server with varying numbers of users, each with keepalive connections. A new user visited the site, and the transaction time and transaction rate the new user achieved was measured.

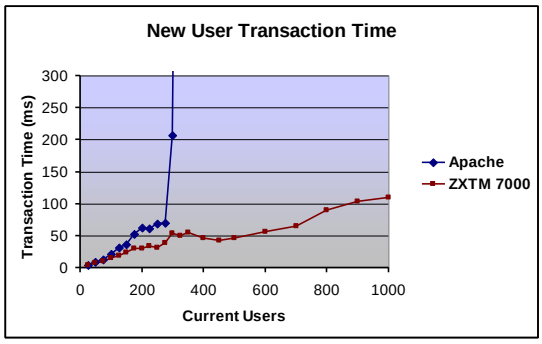




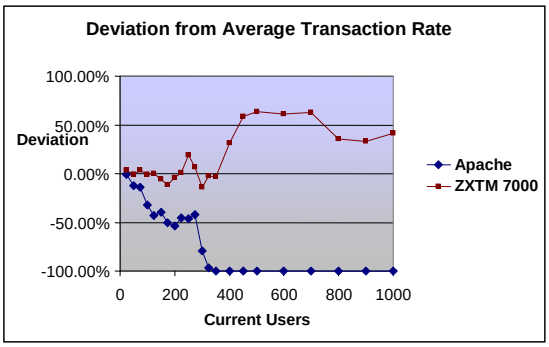
Once over 250 concurrency slots are occupied by site users, the service for additional visitors degrades very rapidly. Existing site users hog all the available resources.

Zeus Traffic Manager was used to transparently manage the client keepalive connections and marshal them into a far smaller number of keepalive connections to the Apache server.





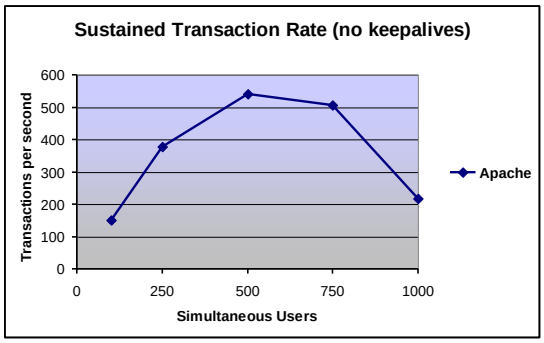
Zeus Traffic Manager ensured that all site users obtained consistent performance and many more simultaneous site visitors could be sustained. It's possible to calculate the deviation from the average transaction rate that the new user experiences:

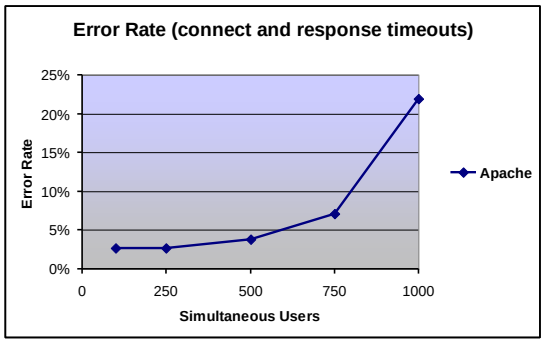


Apache's uneven distribution of service comes as a result of its limited per-thread or per-process concurrency model. Many sites overcome it by disabling or restricting HTTP Keepalives, although this results in poorer overall performance.

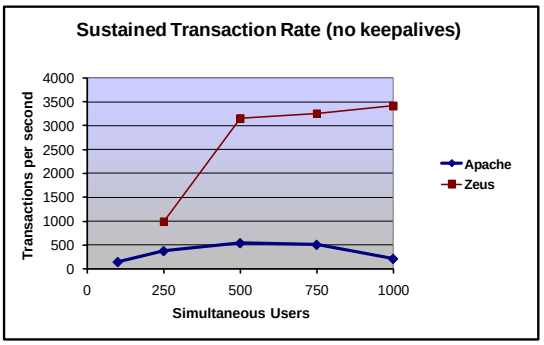
Key Finding 2: Apache's performance improved with Zeus Traffic Manager

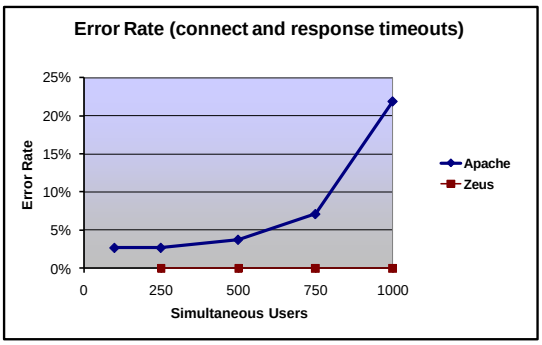
Disabling keepalives on Apache results in poorer overall performance. Broadband-Testing used Spirent Avalanche clients to simulate varying numbers of users accessing the Apache server.





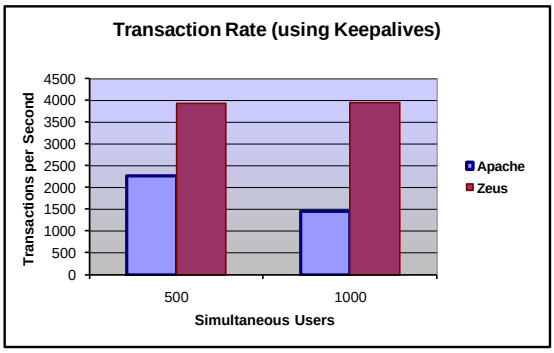
Zeus Traffic Manager was then used to manage and marshal the HTTP requests to the Apache server:





Without keepalives, Zeus lifts the performance of the Apache server dramatically.

The previous finding indicated that it is safe to enable Keepalives with Zeus Traffic Manager, but not with Apache. Nevertheless, the above test was repeated using Keepalives on both systems:

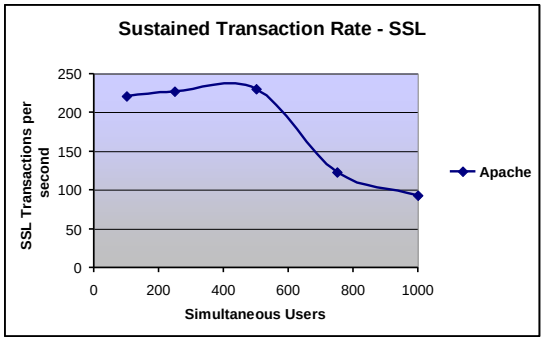


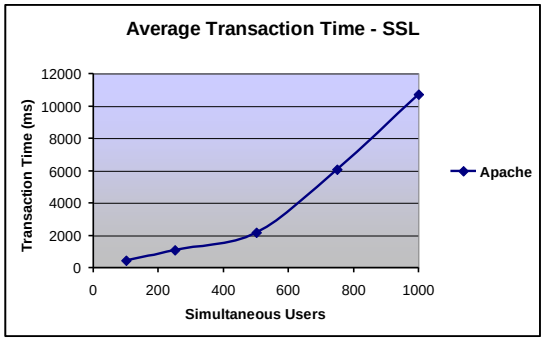
Using Keepalives with Zeus Traffic Manager adds a further performance boost. Note that the Apache server was the bottleneck in all these tests, running at between 90% and 100% CPU utilization. The Zeus Traffic Manager had ample spare capacity to load-balance traffic across a number of Apache servers.

Key Finding 3: Accelerating SSL transactions on Apache server

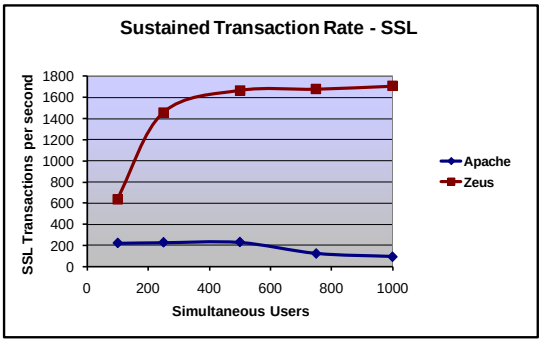
SSL is very processor intensive and puts a limit on the capacity of a web site or service. Broadband-Testing investigated Apache's SSL performance, measuring the performance of SSL transactions by requesting 16 byte files with no SSL session reuse.

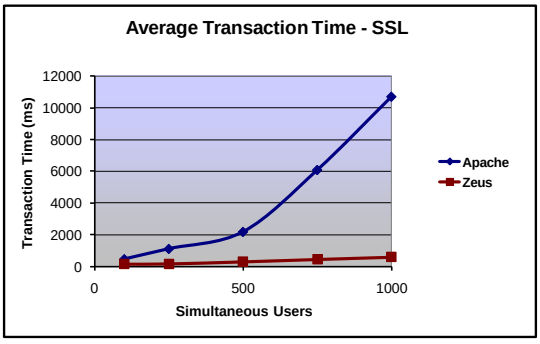
The Apache server was tested directly. CPU utilization consistently reached 100% in these tests, but the server became overloaded with more than 500 simultaneous users, processing transactions increasingly slowly.



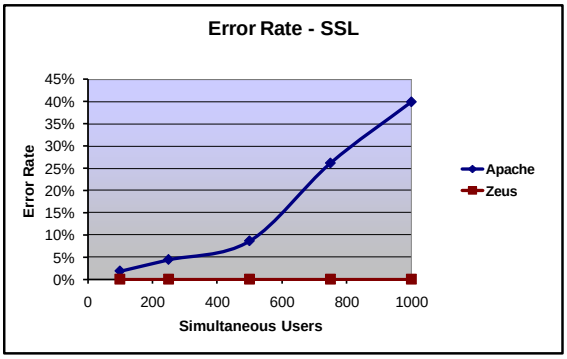


Then Zeus Traffic Manager was used to transparently decrypt SSL traffic on behalf of the Apache server, so that it processed unencrypted traffic only.



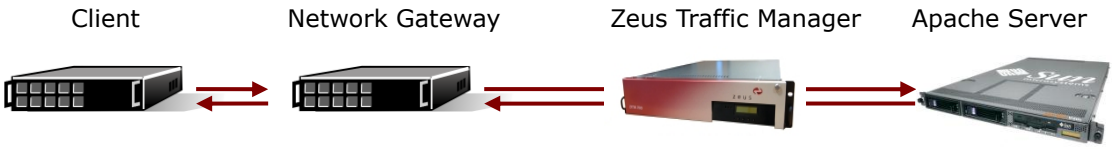


During the tests, many of the connections to the Apache server failed due to timeouts. When Zeus Traffic Manager was used to decrypt the connections, no failures were observed:

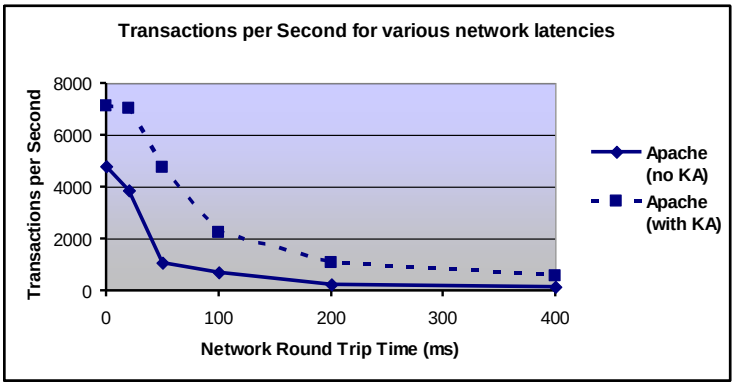


Key Finding 4: Poor performance on high-latency networks

The majority of benchmarks are conducted on fast, local networks, and can conceal problems apparent to any user accessing a site over a remote, high-latency network. This test used 'zeusbench' and an intermediate device[1] to simulate different network conditions, and investigated the performance of Apache (with and without keepalives):



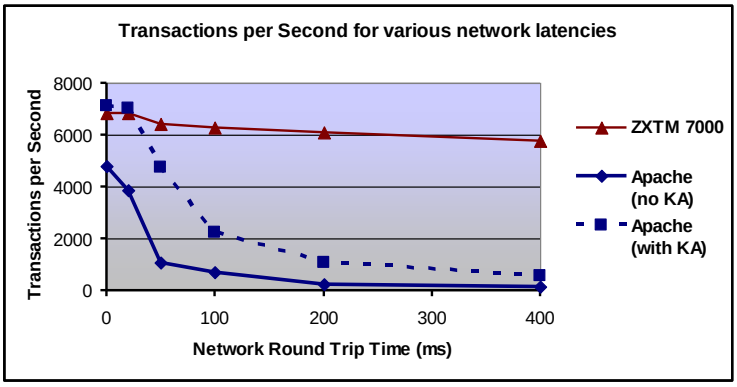
Apache performs poorly on high-latency networks:



Network Round Trip Times	
0 ms	Local Area Network
50 ms	Local Regional Network
100 ms	National Network
200 ms	International Network
400 ms	Intercontinental Network

At high latency values, the Apache server ran at no more than 8% CPU. Even though 5000 concurrent connections were competing for service and the Apache server had plenty of capacity, the latency effects meant that Apache was unable to serve them.

Zeus Traffic Manager can manage the client connections on behalf of Apache. In this case, the Apache server behaves as if it were communicating on a fast, low-latency network.



Zeus Traffic Manager can give consistent performance and can eliminate the latency effects that cause poor performance in Apache.

[1] A Linux gateway server running the Netem network emulation module.

Analysis: How do Keepalives affect Levels of Service?

HTTP Keepalives produce a much better end-user browsing experience, but the Apache Web Server disables them by default because they cause performance problems.

Why use HTTP Keepalives?

The HTTP protocol sends a request using a TCP connection. HTTP *Keepalives* allows a web browser to reuse a TCP connection, sending a number of HTTP requests (for web pages, images, stylesheets and other resources) down the same connection. TCP connections take time to set up and tear down, and without Keepalives, a new TCP connection must be created for each resource on a web page.

All modern web browsers support keepalives (and some also support pipelines within keepalives). The page load time and bandwidth usage benefits of Keepalives and Pipelines are well known[2,3,4,5].

> *Analogy: A telephone conversation uses the equivalent of keepalives. It takes several seconds to set up a call – dialling the number, waiting for the network and for the other party to pick up – and a second or two to close the call.*
>
> *Without Keepalives, you could only speak once on your phone call. The other party would reply and then you would have to hang up. If you wanted to conduct a detailed transaction, or even just a conversation, it would be a slow and laborious process as you would have to hang up and redial between sentences.*
>
> *With Keepalives, the phone call can last as long as you need. You can speak as many times as you wish, and conversation (or transactions) are conducted with ease.*

Pipelines are a further development of keepalives, whereby the web client can issue several requests without having to wait for each reply in turn – just as in a phone conversation, you may ask several questions in one sentence.

Why not use Keepalives with Apache?

The Apache server is not designed to handle Keepalives efficiently. The problem arises because the Apache Server has a relatively low limit on the number of TCP connections ('slots') that it can manage at any one time. Each Keepalive connection occupies one of these slots for up to 15 seconds (the default timeout).

> *A slot corresponds to a process or thread, depending on whether Apache uses its prefork (process) or worker (thread) multiprocessing module. The typical limit is 256 slots for prefork and 150 for worker, and it is generally unwise to configure a larger limit for stability reasons. All tests in this report used prefork because this is the common default and is the most stable and supported MPM.*

[2] http://www.mozilla.org/projects/netlib/http/pipelining-faq.html
[3] http://www.w3.org/Talks/970917HTTP/HTTPSigcomm97-2.ppt
[4] http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/e7b5b10a-5eac-485f-80f0-0e04eaf6c3ba.mspx?mfr=true
[5] http://www.microsoft.com/technet/technetmag/issues/2005/11/PumpUpPerformance/default.aspx

1. Keepalives reduce Apache performance

   An occupied slot takes up resources on the server machine, even if it is occupied by an idle Keepalive connection. In a live deployment, it is more efficient to disable Keepalives and take the hit of accepting and closing many TCP connections rather then to accept the cost of maintaining the many idle keepalive connections.

2. Keepalives give very uneven levels of service in Apache

   If all of the available 'slots' are occupied by active users or idle keepalive connections, there are no free slots for additional users.

   A new user's connection will be queued in the operating system's listen queue, and will not be serviced until an existing idle keepalive connection times out. During times of high load, there is a lot of contention for free slots as they become available. Once the listen queue fills, clients will receive 'connection refused' errors, and connections in the listen queue may time out of they are not serviced quickly.

   New users will get very poor service, while users with existing keepalive connections will be able to reuse them and get good service.

For this reason, Keepalives are commonly disabled by default in Apache. The Apache Software Foundation has implemented an experimental 'event' MPM in Apache 2.2 to cope with the 'keep alive problem'[6]; at the time of writing, this module is experimental, not available on Apache 2.0.x, and incompatible with other Apache features including SSL support.

Keepalives may give Misleading Benchmarks

Keepalives appear to dramatically improve the performance of an Apache server. Using the apachebench tool against an Apache Server on a two-processor SunFire v20Z server, it's easy to fully load up the web server:

	Command Line	Transactions/second	CPU utilisation
With Keepalives	ab –c 300 –t 10 –k	7203	100%
Without Keepalives	ab –c 300 –t 10	5056	70%

Misleading benchmarks indicate good Keepalive performance

These benchmark figures are misleading because they only report completed connections. Once the concurrency figure exceeds the number of slots, additional requests will never be served, so additional visitors to the site will have to wait in a queue until sufficient keepalive connections are closed. Apachebench does not reveal this poor "client experience".

The first benchmark test in this report is designed to identify and measure this phenomenon.

[6] http://httpd.apache.org/docs/2.2/mod/event.html

Benchmarking Apache

The default apache configuration on a RedHat Enterprise Linux 4 system was used, with a 10kB response file. The default prefork MPM was used, with default settings:

StartServers	8	ServerLimit	256
MinSpareServers	5	MaxClients	256
MaxSpareServers	20	MaxRequestsPerChild	4000

Existing users with Keepalive connections were simulated by one apachebench run using various degrees of concurrency:

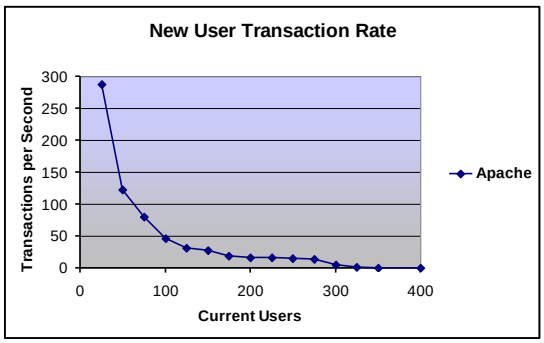
```
ab -c <num_users> -n 10000000 -k http://10.2.2.2/
```

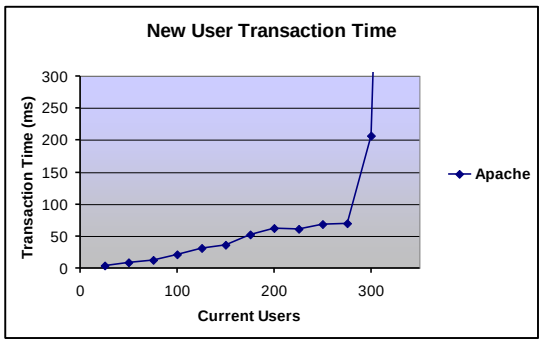
After 15 seconds, a new user attempted to download content from the new server using a new Keepalive connection:

```
ab -c 1 -t 60 -k http://10.2.2.2/
```

The test measured the transaction rate (transactions per second) and transaction time that the user achieved over a 60-second period.

Users:	25	50	75	100	125	150	175	200	225	250	275	300	325
TPS:	288	122	80.0	46.7	31.6	27.6	19.2	16.0	16.4	14.5	14.2	4.84	0.70
Time (ms):	3.47	8.17	12.5	21.4	31.7	36.3	52.0	62.4	61.0	69.0	70.2	206	1430





For more than 325 users, no requests were processed for the additional user.

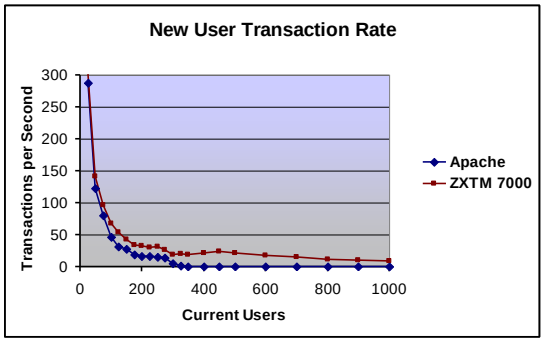
There was a clear degradation of performance once the `MaxClients` limit was exceeded. Some Keepalives were recycled because established connections are closed by the server when the `MaxRequestsPerChild` limit is exceeded. When there was little contention for these additional slots, the new user was able to submit some requests. This explains why requests were processed at the 275 and 300 user level.

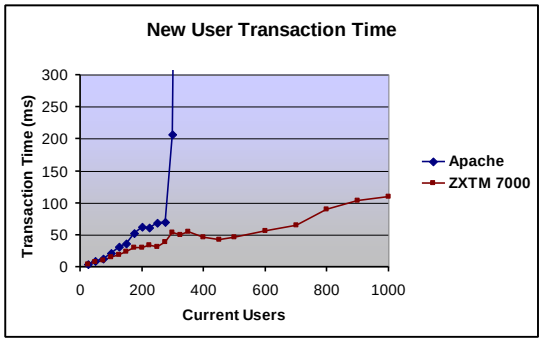
Benchmarking Apache with Zeus Traffic Manager

The test was then repeated, using Zeus Traffic Manager to front-end the Apache server. No configuration changes were made to the Apache Server, and the Zeus Traffic Manager accepted Keepalive connections from the clients.

Users:	25	50	75	100	125	150	175	200	225	250	275
TPS:	300	141	96.4	68.0	54.3	42.3	33.7	32.9	30.1	31.8	25.7
Time (ms):	3.33	7.11	10.4	14.7	18.4	23.7	29.7	30.4	33.3	31.4	39.0

Users:	300	325	350	400	450	500	600	700	800	900	1000
TPS:	18.7	20.3	18.4	21.8	23.4	21.7	17.7	15.4	11.1	9.63	9.14
Time (ms):	53.4	49.2	54.2	45.8	42.8	46.0	56.5	64.7	90.1	103	109





Zeus Traffic Manager ensured that all site users obtained consistent performance over a very wide range of user numbers, and that many more simultaneous site visitors could be sustained.

The results clearly illustrate how Zeus Traffic Manager can support many more Keepalive users than Apache, giving better, consistent levels of service to all users.

Analysis: Accelerating Apache with Zeus Traffic Manager

Zeus Traffic Manager functions as an application proxy, managing client side connections and maintaining shared server-side keepalive channels. It is capable of reducing hundreds of thousands of simultaneous client connections into a much smaller number of server keepalive connections.

It was hypothesized that Zeus Traffic Manager could manage keepalive connections on behalf of the Apache server to deliver better, more even performance.

Broadband-Testing performed a number of tests to ascertain whether Zeus Traffic Manager could effectively accelerate Apache servers. Broadband-Testing used Spirent Avalanche load generators to perform the tests, and Spirent software to collect and analyse the results.

Testing Zeus Traffic Manager's Acceleration Benefits

Without Keepalives

The Spirent Avalanche clients were configured to simulate various numbers of users, each of whom repeatedly requested the 10k file. The clients did not request Keepalive connections.

Sim. Users	TPS (peak)	TPS (average)	Error Rate	Time
250	474	378	2.71%	530 ms
500	937	541	3.77%	740 ms
750	1280	507	7.12%	1180 ms
1000	835	217	21.89%	3690 ms

Apache Performance

The test was repeated, using Zeus Traffic Manager to accept the connections from the clients. Zeus Traffic Manager maintained a small set of keepalive connections to the Apache server:

Sim. Users	TPS (peak)	TPS (average)	Error Rate	Time
250	1250	997	0.00%	201 ms
500	3540	3150	0.00%	127 ms
750	3570	3250	0.00%	185 ms
1000	3670	3410	0.00%	234 ms

Zeus Traffic Manager + Apache performance

The results show that Zeus Traffic Manager is significantly more efficient at handling non-keepalive connections, translating these into efficient, managed keepalive connections that keep the Apache server operating at peak performance. Zeus Traffic Manager never exceeded 20% CPU utilisation, whereas the Apache server was fully utilised in all tests at 500 simultaneous users and above.

Apache's performance was particularly bursty and uneven, as evidenced by the large discrepancy between the peak and average transaction-per-second figures.

With Keepalives

Even when using Keepalives, Zeus Traffic Manager still helped the Apache server perform more rapidly, more evenly and in a more stable manner:

Sim. Users	TPS (peak)	TPS (average)	Error Rate	Time
500	3490	2260	1.33%	220 ms
1000	3890	1460	5.32%	687 ms

Apache Performance with Keepalives

Sim. Users	TPS (peak)	TPS (average)	Error Rate	Time
500	3980	3630	0.00%	138 ms
1000	4010	3940	0.04%	254 ms

Zeus Traffic Manager + Apache Performance with Keepalives

Why does Zeus Traffic Manager perform better?

Apache's Heavyweight Thread/Process Architecture

Apache's prefork or worker multi-processing modules create an entire operating system thread or process for each individual connection the server is managing. This is an excessively heavyweight architecture as a thread or process consumes many more resources than the single connection.

The architecture causes severe scalability problems when managing many connections. The server OS spends significant time housekeeping the hundreds of thread or processes, and its general purpose thread and processing scheduling algorithms are not optimised or specialised for simple scheduling based on network IO.

Connection processing blocks frequently when the thread or process waits for network data, buffer space or TCP closes; each time, the operating system must them perform a heavyweight context switch so that it can process another connection.

Zeus Traffic Manager's Lightweight Non-Blocking Architecture

Zeus Traffic Manager's architecture scales with the concurrency of the host hardware. Zeus Traffic Manager runs a single process for each processing unit (core or processor). Each process is capable of managing many thousands of connections simultaneously, switching between them using the OS 'epoll' system call.

This architecture is commonly described as 'select-based', because many implementations use the 'select' system call to inspect many connections and determine which can be processed without blocking; 'epoll' is a more efficient and scalable version of 'select' when inspecting large numbers of connections.

Analysis: Apache's SSL Performance

SSL-offload devices can dramatically improve Apache's performance

SSL performance can be a limiting factor for many sites. Netcraft reported that some banks are shifting logins to non-SSL pages[7], compromising security for performance reasons.

The SSL Alternatives

Apache's SSL implementation is based on a freely available open source cryptographic library. Several proprietary, commercial libraries are available which offer significantly higher performance; these libraries are not used in Apache for commercial and licensing reasons.

Apache can be used with commercial SSL accelerator hardware, or with a specialised load balancer such as Zeus Traffic Manager that can decrypt traffic at a fast rate. If the second option is chosen, then the performance of Apache is dramatically improved to the level achieved when processing plaintext (unencrypted) protocols.

Testing Apache's SSL Performance

Broadband-Testing evaluated Apache's SSL performance, with and without Zeus Traffic Manager. Two Spirent Avalanche clients simulated varying numbers of users requesting a 10k file from the Apache Server, without any keepalives. The Apache server used a 1024 bit RSA private key; RC4 and MD5 were used in the domestic-strength SSL cipher:

Sim. Users	TPS (peak)	TPS (average)	Error Rate	Time
250	233	227	4.42%	1100 ms
500	229	230	8.61%	2175 ms
750	228	123	26.1%	6085 ms
1000	228	93	39.9%	10715 ms

Apache Performance

Apache ran at a consistent 100% CPU utilization during these tests.

The test was repeated using Zeus Traffic Manager to accept and decrypt the client connections (again with a 1024 bit RSA key and the same SSL ciphers). No keepalives were used.

Sim. Users	RPS (peak)	RPS (average)	Error Rate	Time
250	2070	1460	0.00%	172 ms
500	1970	1670	0.00%	300 ms
750	2140	1680	0.00%	447 ms
1000	2150	1710	0.00%	586 ms

Zeus Traffic Manager + Apache Performance

[7] http://news.netcraft.com/archives/2005/08/23/banks_shifting_logins_to_nonssl_pages.html

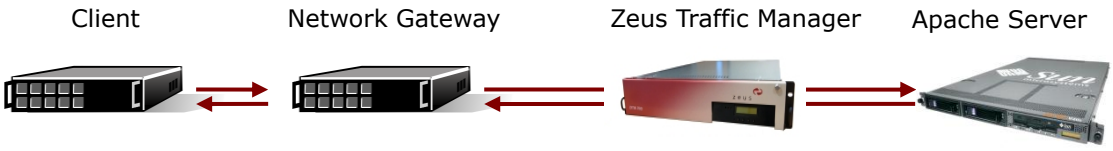
Analysis: High-latency Networks

Apache performs particularly poorly on high-latency networks.

The vast majority of web browsing occurs over wide-area networks where latency is high. However, the vast majority of performance testing is conducted on local networks where latency is low. This testing fails to identify problems that are exacerbated by high-latency networks.

Test Methodology

The final test in this report used an apachebench-like tool ('zeusbench') to simulate 5000 remote users accessing an Apache server, each requesting 10k files as quickly as possible. A Linux gateway running the Netem[8] Network Emulation module was used to simulate different network latencies.



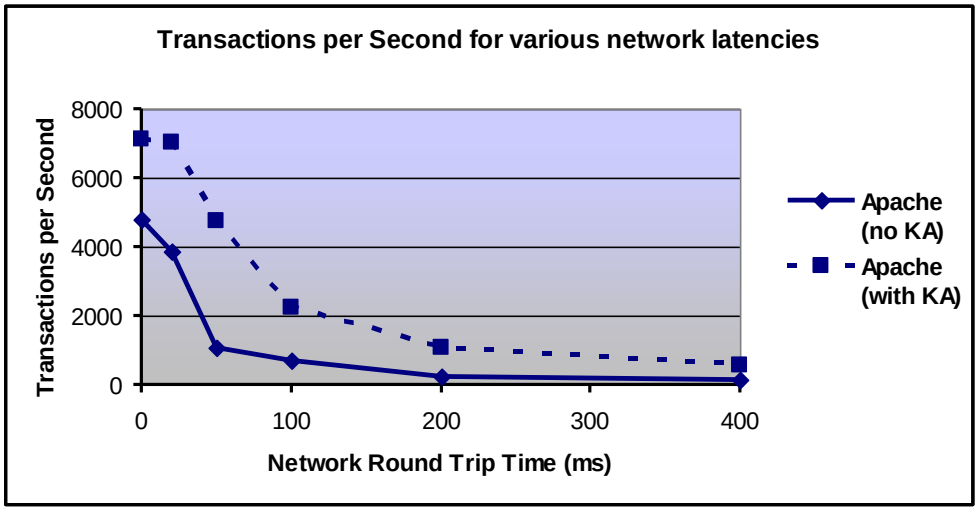
The Netem module creates a queue for a network interface, delaying packets that enter that interface for a configured period. For example, to configure a round trip time of 100ms on the gateway machine, the incoming and outgoing interfaces each needed a delay of 50 ms:

```
# tc qdisc add dev <interface> root netem delay 50ms limit 100000
```

Network round trip times could be easily verified with 'ping'.

Testing Apache

Apache was run on a 2-processor Opteron 244 server[9]. It was tested with and without keepalives:



[8] http://linux-net.osdl.org/index.php/Netem
[9] For comparison purposes, the previous Broadband Testing tests used a single-processor equivalent, so total TSP figures were lower.

Network RTT[10]	0 ms	20 ms	50 ms	100 ms	200 ms	400 ms
TPS (no keepalives)	4800	3850	1080	700	253	162
TPS (with keepalives)	7110	7020	4740	2250	1080	536

Apache transactions per second

As the network round-trip-time increased, connection read, write and close operations all took longer. Consequently, each connection occupied a process slot for a longer period.

At high latency values, the Apache server ran at approximately 8% CPU. Even though 5000 concurrent connections were competing for service and the Apache server had plenty of spare capacity, the latency effects meant that Apache was unable to serve them.

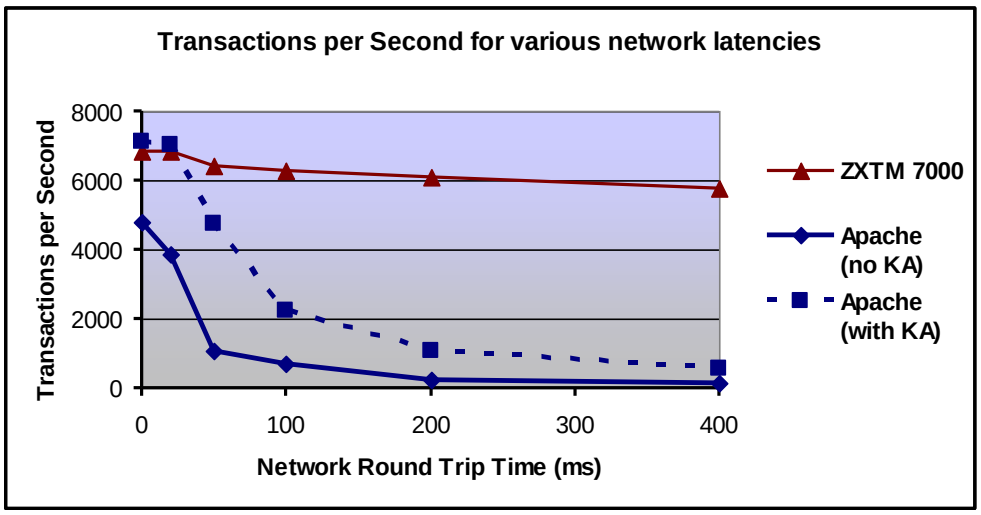
In limited-concurrency architectures like Apache's, the maximum rate at which new connections can be accepted is limited by the following quantity:

Number of connection slots / total duration of each connection

Adding more CPU capacity, memory or network bandwidth has little or no effect on this scalability limit. The key to improving the performance design such is to reduce the duration of each connection.

Improving performance with Zeus Traffic Manager

Zeus Traffic Manager was used to accept connections on Apache's behalf and forward them to the Apache server. From the Apache server's perspective, the connection appears to originate from the local network, so latency effects are completely eliminated. As the results show, Zeus Traffic Manager is much more effective at managing large numbers of very slow connections.



[10] Latency is measured in terms of the round-trip time (RTT). Empirical testing suggests that for 10k file transfers with no contention, connections last for 3-4 times the RTT for non-keepalive connections, and 1-1.5 times RTT for keepalive connections. Connections last significantly longer with contention.

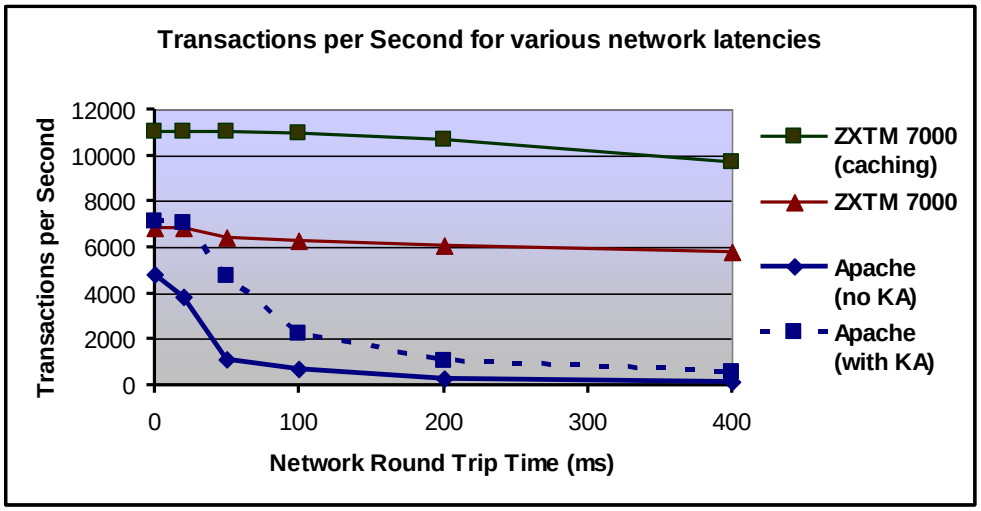
Network RTT	0 ms	20 ms	50 ms	100 ms	200 ms	400 ms
TPS (with keepalives)	6830	6860	6400	6280	6070	5780

Zeus Traffic Manager + Apache transactions per second

The Apache server was the bottleneck in the first tests, running at or near 100% CPU. Zeus Traffic Manager was unable to drive it any faster. Across all latency values, Zeus Traffic Manager was able to accelerate Apache to near its peak capacity running in the optimal low-latency environment.

Using Zeus Traffic Manager's Content Caching

Zeus Traffic Manager's Content Caching capability offloads the majority of HTTP transactions from the Apache server. When Content Caching is enabled, even on high latency networks, it is possible to effectively utilise all the available bandwidth[11]:



Network RTT	0 ms	20 ms	50 ms	100 ms	200 ms	400 ms
TPS (inc. content cache)	11100	11100	11000	10900	10700	9720

In this 'content cache' test, the Zeus Traffic Manager served the 10kB responses from its internal cache, only querying the Apache server periodically to ensure the cached content was accurate. In this case, the 1Gbits network was the bottleneck for the majority of the tests, and imposed a practical limit of just over 11000 TPS.

[11] Max transaction rate, allowing for 15% HTTP and IP overhead, is 11400 transactions/second.

About Broadband-Testing

Broadband-Testing is Europe's foremost independent network testing facility and consultancy organisation for broadband and network infrastructure products.

Based in the south of France, Broadband-Testing offers extensive labs, demo and conference facilities. From this base, Broadband-Testing provides a range of specialist IT, networking and development services to vendors and end-user organisations throughout Europe, SEAP and the United States.

Broadband-Testing is an associate of the following:

- NSS Network Testing Laboratories (specialising in security product testing)
- Broadband Vantage (broadband consultancy group)
- Limbo Creatives (bespoke software development)

Broadband-Testing Laboratories are available to vendors and end-users for fully independent testing of networking, communications and security hardware and software.

Broadband-Testing Laboratories operates an Approval scheme which enables products to be short-listed for purchase by end-users, based on their successful approval.

Output from the labs, including detailed research reports, articles and white papers on the latest network-related technologies, are made available free of charge on our web site at http://www.broadband-testing.co.uk

The conference centre in Moux in the south of France is the ideal location for sales training, general seminars and product launches, and Broadband-Testing can also provide technical writing services for sales, marketing and technical documentation, as well as documentation and test-house facilities for product development.

Broadband-Testing Consultancy Services offers a range of network consultancy services including network design, strategy planning, Internet connectivity and product development assistance.

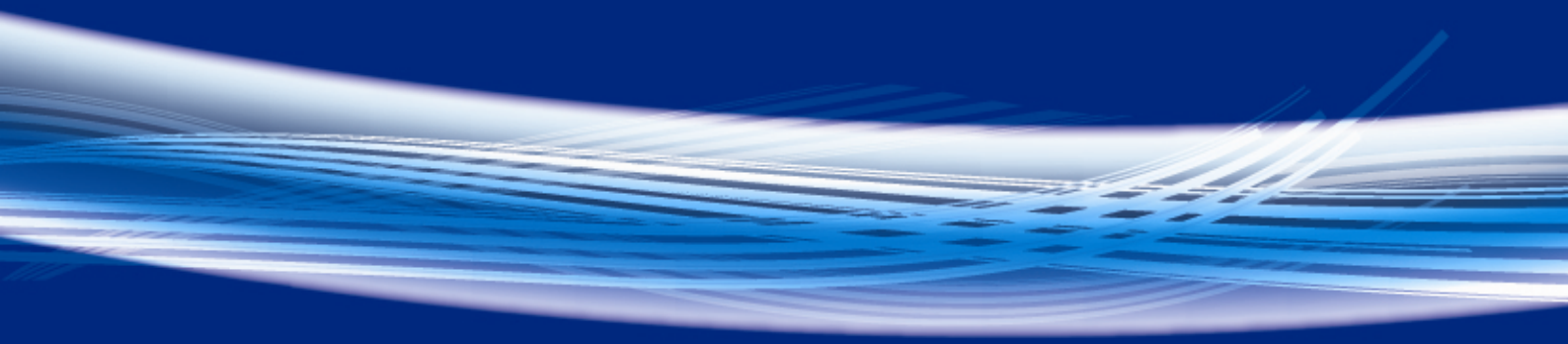
About Zeus Technology

Zeus software enables our customers to create, manage and deliver exceptional online services in Physical, Virtual and Cloud environments. Implementing a Zeus solution allows organizations to visualize and manipulate the flow of traffic to their web-enabled applications, thus ensuring a consistently robust web infrastructure.

Coupled with the ability to deploy new online services very quickly, Zeus helps provide the competitive advantage businesses need, by making their online services faster, more reliable, more secure and easier to manage. Zeus software can be deployed on industry standard hardware, virtual machines and any Cloud platform, making it the right strategic choice for today and tomorrow.

Zeus holds strategic partnerships with world-class companies such as Dell, HP, IBM, Intel, Joyent, Oracle, Sun, Qualcomm and VMware. Zeus powers over one million website infrastructures across the world including, BT, Comic Relief, Domino's Pizza, Gilt Groupe, ITV, PLAY.COM, STA Travel and Virgin Media.

Zeus is in a unique position to assist any business looking to enhance their mission-critical Internet service infrastructure.



For further information, please email: info@zeus.com or visit www.zeus.com
Stay in touch with Zeus by following: blog.zeus.com or twitter.com/ZeusTechnology

Try before you buy.
Simply visit our website: www.zeus.com/downloads
Technical support is also available during your evaluation.

Zeus Technology Limited (UK)
The Jeffreys Building
Cowley Road
Cambridge CB4 0WS
United Kingdom

Sales: +44 (0)1223 568555
Main: +44 (0)1223 525000
Fax: +44 (0)1223 525100
Email: info@zeus.com
Web: www.zeus.com

Zeus Technology, Inc. (U.S.)
1875 South Grant Street
Suite 720
San Mateo, California 94402
United States of America.

Phone: 1-888-ZEUS-INC
Fax: 1-866-628-7884
Email: info@zeus.com
Web: www.zeus.com

zeus