

z e u s



Accelerating Web Applications with ZXTM

Delivering better performance, stability and return-on-investment from web-enabled applications.

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

Zeus Technology
1955 Landings Drive
Mountain View
CA 94043
United States

UK: +44 (0)1223 525000
US: +1 650 965 4627

Email: info@zeus.com
Web: <http://www.zeus.com/>

Contents

Introduction	3
What performance problems are incurred?.....	3
What is the solution?	3
ZXTM Application Acceleration: Key Facts	3
Reasons for Poor Performance	4
Reason 1: Out of the application’s comfort zone.....	4
Reason 2: The Concurrency Model	4
How much concurrency is needed?	4
Why is there a concurrency limit?	5
What is the effect of the concurrency limit?	5
1. Restricting HTTP Keepalives	6
2. Fewer Simultaneous Users	6
3. Poor performance on a WAN	7
Solving the concurrency problems	7
How does ZXTM help?	8
Offloading Operations	8
SSL Transactions.....	8
Content Compression.....	8
XML Operations.....	9
Content Caching	9
Managing client-side connections.....	9
Conclusion	11
Customer Testimonials	11



Introduction

Zeus Extensible Traffic Manager (ZXTM) is a software load balancer for networked and web-enabled applications. It improves the performance, reliability and security of these applications, and reduces operational costs across complex, multi-tiered and fragile infrastructures.

What performance problems are incurred?

Many common web application platforms suffer severe performance problems. Their workload gives them a range of tasks they are not optimized for; they scale poorly when handling large numbers of clients; they under-perform with connections over slow, high-latency networks.

These problems are particularly common with thread- or process-based server applications, such as the Apache Web Server, and many Java-based application servers. They are exacerbated further by software virtualizations such as VMware and Xen that add additional networking layers.

What is the solution?

Various 'point' solutions address some aspects of the performance problems; for example, SSL or XML accelerators offload some of the processing tasks onto specialized hardware.

ZXTM provides a complete solution for all of the performance problems. This white paper describes these problems in detail, and explains how ZXTM is able to solve them.

For independent validation of the performance benefits described in this document, refer to the BroadBand Testing reports published at <http://www.zeus.com/products/zxtm/>.



The ZXTM software is based on the proven Zeus Web Server product. ZWS is the server of choice for hardware manufacturers such as AMD, Intel, HP, Sun and IBM seeking to get the very most from their servers in the industry-standard SPECweb performance tests.

ZXTM Application Acceleration: Key Facts

ZXTM accelerates web and networked applications by:

- Offloading compute intensive tasks – SSL decryption, Content Compression and XML processing – onto high-performance software optimized for low-cost 64 bit processors.
- Caching commonly requested data from the server to reduce server load.
- Offloading and buffering slow TCP client connections so that the server only communicates over a fast local network.
- Multiplexing HTTP requests so that the server only communicates with a small number of very efficient clients.



Reasons for Poor Performance

Poor performance arises because server applications perform tasks they are not optimized for, and because many server applications use a concurrency model that is easy to program but scales badly.

Reason 1: Out of the application's comfort zone

A Java application server is designed for one key purpose – to manage and run application code in a Java environment. The Apache server is designed to be as portable as possible across many hardware and OS platforms, and to be feature-rich to support a great many third-party application modules.

Performance is a secondary concern in the design of these servers, and few development resources are concentrated in addressing the performance problems in compute-intensive ancillary tasks such as SSL decryption, compressing response data or performing XML operations.

In some cases, the application design makes it impossible to select the best possible implementation. Java applications are reliant on Java-based implementations of computing tasks; Apache uses lower-performance open source cryptographic libraries rather than the higher performance commercial ones that are available.

A dedicated acceleration device which performs these compute intensive tasks can bring great performance benefits. For example, the Apache server processes SSL-encrypted traffic between 10 and 20 times more slowly than non-encrypted traffic, so offloading the SSL-decryption operation onto a separate device will increase capacity by 10 or 20 times.

The return-on-investment is very clear when you consider commercial application servers that are priced on a per-CPU basis. In this case, it's vital to offload all unnecessary processing from the application server, in order to gain the best possible return on the per-CPU licensing cost.

Reason 2: The Concurrency Model

Concurrency is the number of simultaneous connections that a server can handle.

The *Concurrency Model* used in the design of a server application greatly influences how the application processes many simultaneous connections.

The limited Concurrency Model used by server applications like Apache and many Java Application Servers is the biggest cause of the performance problems they experience.

How much concurrency is needed?

A typical web client like Internet Explorer or Mozilla FireFox makes several simultaneous connections to a web server to download the content, and holds these connections open for 15 seconds (the 'keepalive timeout') afterwards in case they need to be reused.



Suppose your web server can handle 256 concurrent connections (a common limitation in Apache). Each client uses 2 connections to download a web page, and each connection is open for slightly more than 15 seconds. Then you are immediately limited to 8 users per second on your server:

$$\text{transactions per second} = \frac{\text{concurrency limit}}{2 \times \text{connection duration}} = \frac{256}{2 \times 16} = 8 \text{ transactions per second}$$

Why is there a concurrency limit?

Web servers like Apache, and the majority of Java Application servers use a simple design to cope with concurrent connections. They use a separate *process* or *thread* to handle each connection.

This design lends itself to straightforward, reliable code because each connection is handled in its own, isolated environment. It is very suitable for application servers and servers like Apache which run third-party code because it has a much simpler programming model. The developer can write application code that performs blocking operations (such as a database transaction) without needing to concern himself with interactions between other connections that the server is managing.

However, the design is inefficient because it puts a significant load on the underlying operating system. Compared to a simple network connection, a process or thread is a very heavyweight operating system object that consumes many more resources than the connection itself.

The operating system experiences severe scalability problems when managing a large number of concurrent processes or threads. It spends a large proportion of its time housekeeping the hundreds of thread or processes (this proportion grows polynomially), and fewer and fewer CPU resources are left to run the web server or application server code, so the capacity of the system drops and the response time is affected.

A runaway web server can easily overwhelm a machine with so many threads or processes that it becomes completely unresponsive. For this reason, the Apache server is artificially limited to 256 concurrent connections¹. Java-based application servers have similar limitations.

Note: *The Apache Software Foundation has implemented an experimental 'event' MPM in Apache 2.2 to cope with their 'keep alive problem'²; 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.*

What is the effect of the concurrency limit?

The concurrency limit imposes three restrictions on the performance of the web or application server:

¹ 256 for the 'prefork' MPM; 150 for the 'worker' MPM. Increasing this value will reduce the stability of the Apache system

² <http://httpd.apache.org/docs/2.2/mod/event.html>



I. Restricting HTTP Keepalives

HTTP Keepalives produce a much better end-user browsing experience because they make a web site much more responsive. However, concurrency-limited servers like Apache Web Server disable them by default because they can reserve the limited concurrency 'slots' for too long.

For maximum benefit, keepalives should persist for the period of time that a user views a web page, so that when the user clicks on a new link, the TCP connection to download the new content is already established. AJAX applications may need even longer keepalives in order to remain responsive.

However, the majority of high traffic Apache sites either disable Keepalives completely, or reduce the timeout to less than 5 seconds.

Recall the calculation used to determine the maximum number of users to a server:

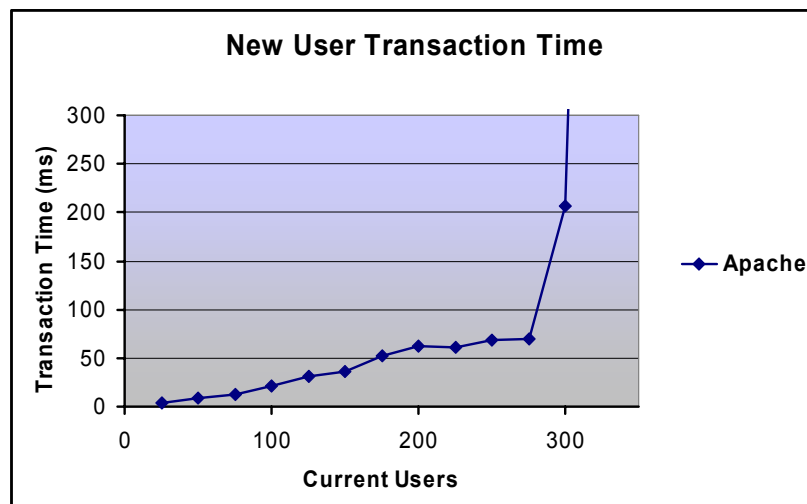
$$\text{transactions per second} = \frac{\text{concurrency limit}}{2 \times \text{connection duration}}$$

The concurrency for an apache server is 256 (when using the *worker* MPM, it is 150); 2 is the number of simultaneous connections a typical client makes. The only control an administrator has is to reduce the connection duration by limiting keepalives.

2. Fewer Simultaneous Users

The concurrency limit puts a fixed upper limit on the number of users. Additional users are locked out; they won't be able to access the service at all until an existing user's connection completes.

The following graph shows the level of service that a new user experiences when he tries to access an Apache server that is already heavily used by up to 300 current users.

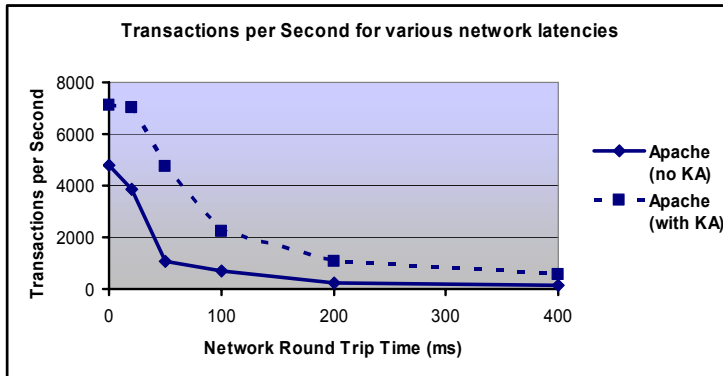


Once there are more users than the concurrency limit can handle, the level of service for new users becomes very poor. The existing users acquire the concurrency slots, and requests from new users do not get processed until an existing user relinquishes his slot.



3. Poor performance on a WAN

Slow, high latency networks generate much longer-duration connections. These connections spend most of their time idle as they wait for more network data, but they still occupy the limited concurrency slots:



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

Increasing the concurrency by increasing the number of processes or threads only has a limited effect.

Solving the concurrency problems

There are alternative server architectures can be used. Zeus Web Server and ZXTM use a higher-performance 'select-based' architecture which 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 model scales evenly with the concurrency of the host hardware.

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

This architecture is appropriate for high-speed web servers and traffic managers, but is not appropriate for complex application servers that run third party code because the programming model is much more complicated. For example, it is extremely difficult to construct code that must perform blocking operations (such a DNS lookup, or a database transaction) within this architecture.

You can overcome the limitations of the concurrency model by using ZXTM to manage the many slow keepalive connections on behalf of the Apache server or Application Server.



How does ZXTM help?

ZXTM has a range of capabilities that improve the performance of servers and applications that it manages traffic to:

- SSL Decryption
- Content Compression
- XML processing (XSLT transformations, XPath queries)
- Content Caching
- TCP offload and buffering
- HTTP multiplexing
- Performance-sensitive load-balancing

Offloading Operations

SSL Transactions

ZXTM's proven SSL stack is optimized for 64-bit x86 platforms like AMD Opteron and Intel Xeon. ZXTM running on a dual-processor dual-core Opteron 285 machine can decrypt and load-balance over 9000 SSL transactions per second³.

In recent tests⁴, it was demonstrated that:

- Using ZXTM to decrypt SSL traffic to a single Apache server provides up to **20-times the transaction rate** and **20-times faster transactions**, with no connection errors.

The ZXTM 7000 was running at 30% utilization, so could comfortably accelerate three Apache servers at the same rate.

- Using ZXTM to decrypt SSL traffic for WebLogic provided **over 15-times the SSL performance**.

The ZXTM 7000 was running at less than 20% utilization, so could simultaneously manage and decrypt traffic to 5 WebLogic servers if required.

Content Compression

ZXTM can perform on-the-fly content compression, offloading this compute intensive task from the back-end servers. Content compression reduces the bandwidth used by a service by up to 50%, and can improve the response time by a small amount over slow, high-latency networks because it reduces the amount of data that has to be transferred.

³ Figures from benchmark tests on the ZXTM 7400 Appliance

⁴ Visit <http://www.zeus.com/products/zxtm> to access the BroadBand Testing Apache and BEA WebLogic performance white papers.



XML Operations

XML operations are extremely compute intensive and often perform poorly on Java-based servers. ZXTM can perform XML validation, XPath queries and XSLT transformations to request and response data on behalf of an application server.

Published benchmarks demonstrate a ZXTM 7400 Appliance system performing XSLT transformations at over 1 Gbit/s.

Content Caching

The simplest way to accelerate an application is to cache as much as possible of its response data to minimise the number of requests that it must handle.

ZXTM's content caching capability does precisely that, using a high performance memory-based cache that is fully compliant with the recommendations of RFC 2616. The content caching decisions can be controlled by TrafficScript, and ZXTM allows the administrator to define unique cache keys to cache multiple versions of the same content – for example, different home pages depending on whether the user is logged in or not.

ZXTM allows the administrator to create very cost effective content caches. Its software architecture means that it can be run on hardware that is appropriately sized for the cache required, and its 64-bit memory addressing means that cache sizes greater than 2Gb can be easily achieved.

Managing client-side connections

ZXTM functions as a full proxy, managing large numbers of slow, unreliable connections on behalf of the back-end applications.

In the case of HTTP:

1. ZXTM performs the slow TCP connection 'accept' and reads the entire request⁵ before connecting to a back-end server;
2. ZXTM connects to a back end server and writes the client request over the fast local network, reusing an existing server-side Keepalive connection if possible;
3. ZXTM reads the entire response from the back-end server rapidly over the network;
4. ZXTM then either closes the server-side connection (if the server requested it), or holds the keepalive connection open for reuse;
5. ZXTM writes the response back to the remote client over the slow remote network.

The server application operates just as if it were talking to a small number of clients on a fast, local network.

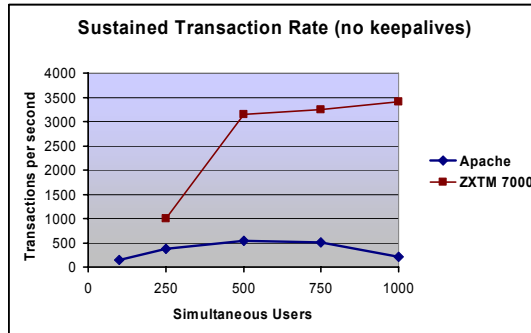
ZXTM manages client side connections completely separately from server side ones. It fully supports all HTTP performance optimizations: keepalive, HTTP pipelining, compression and chunk transfer encoding.

⁵ For large requests such as HTTP Posts, ZXTM will read the request headers and then stream the POST body to the client. This prevents memory starvation on the ZXTM system.

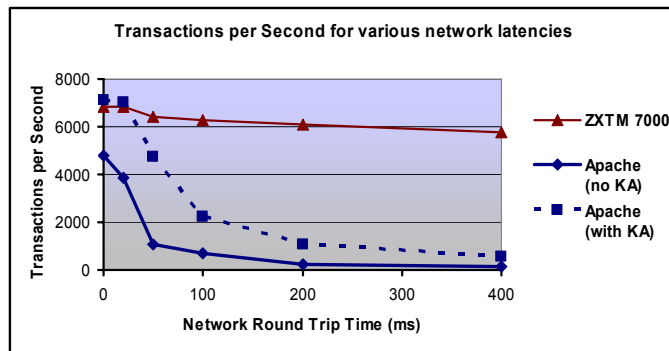


The effects are dramatic:

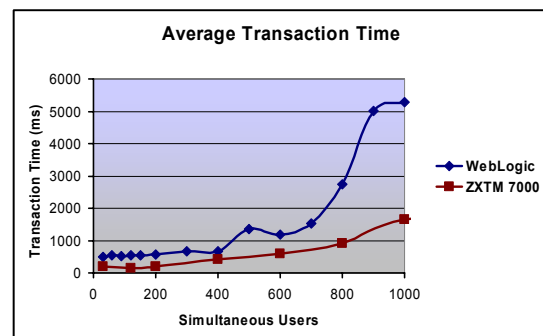
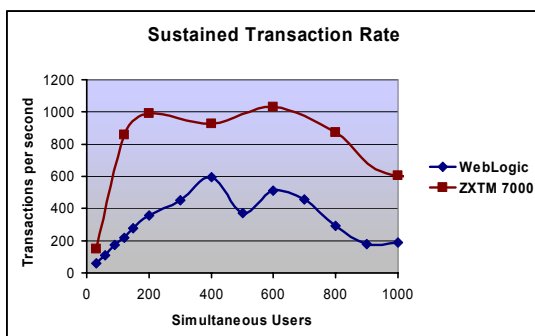
- Using ZXTM to front-end an Apache server results in an increase of up to **18-times the sustained transaction rate**.



- ZXTM almost totally eliminates the high-latency effects, giving up to **40-times better utilisation** and **8-times faster transaction times** when managing traffic to Apache over a 200 ms latency network.



- WebLogic is significantly slower than Apache, so the increased performance hits incurred by large numbers of clients are less acute. Nevertheless, ZXTM still improves the capacity and response times from a WebLogic server:



Conclusion

ZXTM's Application Acceleration results in faster, more responsive and more reliable web sites, with significantly better return-on-investment on application hardware and software.

The benchmarks that illustrate these results are fully documented in the BroadBand Testing white papers that are available from <http://www.zeus.com/products/zxtm/>.

Customer Testimonials

"ZXTM has dramatically improved our web browsing and booking response times"



"We have been extremely impressed with the capabilities of ZXTM. It has

fulfilled all our requirements to create a scalable, resilient and high-performance e-commerce site"

Robin Fitton, Head of IT Operations, Virgin Holidays

700% improvement in application response times



"The real challenge was to maximize existing resources, rather than having to continually add new servers.

"Zeus is leaps and bounds ahead of its competition in all areas of the game."

Kent Wright, Systems Administrator, QuantumMail.com

"It's taken everything we've thrown at it so far"

"Since ZXTM was deployed, there has been a major improvement to the performance and responsiveness of the site."

David Turner, Systems Architect, PLAY.COM



"Really exceeded our expectations"



"Since implementing the solution we have experienced a much better response time from our applications, especially in the handling of slow end user connections."

Torsten Sturm, CTO, Hotel.de

"ZXTM has been rock solid"

"When it comes to flexibility and speed, ZXTM outperforms any other solution we've looked at."

Frank Denis, System Administrator, Skyrock.com



Copyright

© Zeus Technology Limited 2006. Copyright in this document belongs to Zeus Technology Limited. All rights are reserved.

Trademarks

Zeus Technology, the Zeus logo, Zeus Web Server, Zeus Load Balancer, Zeus Extensible Traffic Manager, ZXTM and associated logos and abbreviations, TrafficScript, TrafficCluster and RuleBuilder are trademarks of Zeus Technology Limited. Other trademarks may be owned by third parties.

Contact Information

If you would like to learn more about any of the topics covered by this white paper, please feel free to contact us for more information. You can reach us in a variety of ways:

By Email

For general enquiries:	info@zeus.com
For commercial and technical enquiries:	sales@zeus.com
For reseller information:	partners@zeus.com
For press and public relations information:	press@zeus.com

By Telephone

Zeus Technology UK:	+44 (0)1223 525000
Zeus Technology US:	+1 650 965 4627
Fax:	+44 (0)1223 525100

By Post or in Person

Zeus Technology Limited The Jeffreys Building Cowley Road Cambridge CB4 0WS United Kingdom	Zeus Technology 1955 Landings Drive Mountain View CA 94043 United States
--	--

www.zeus.com

Our web site contains a wealth of information on our products, services and solutions, as well as customer case studies and press information. For more information, please visit <http://www.zeus.com/>.

knowledgehub.zeus.com

The ZXTM KnowledgeHub is a key resource for developers and system administrators wishing to learn about ZXTM and Zeus' Traffic Management solutions. It is located at <http://knowledgehub.zeus.com/>.

