

# Sun™ Secure Application Switch - N2000 Series: SSL and Layer 7 performance test

*Test report prepared under contract from Sun Microsystems*

## Executive summary

Sun Microsystems commissioned VeriTest to measure the SSL and Layer 7 performance of their Sun™ Secure Application Switch - N2000 Series. Sun supplied a N2120 switch running version V2\_0A47336 firmware and configured with 12 small form factor Gigabit Ethernet ports and one function card. The function card adds support for both hardware-based Layer 4-7 application switching and SSL acceleration.

Sun requested that VeriTest measure the following metrics on the Sun N2000 Series switch:

- Maximum SSL connections/second with no SSL session reuse
- Maximum SSL bulk cryptographic throughput with no SSL session reuse
- Maximum Layer 7 connections/second
- Maximum Layer 7 throughput

### Key findings

- ❑ The Sun N2000 Series switch averaged 12,569 SSL connections/second in our test configuration.
- ❑ The Sun N2000 Series switch averaged 2.102 Gigabits/second of encrypted SSL throughput in our test configuration.
- ❑ The Sun N2000 Series switch averaged 56,140 Layer 7 connections/second in our test configuration.
- ❑ The Sun N2000 Series switch averaged 2.293 Gigabits/second of Layer 7 throughput in our test configuration.

For each of the performance tests, the Sun N2000 Series switch was configured as a load balancer and TCP connection termination device positioned between clients generating HTTP requests and Web servers responding to requests.

We used Ziff Davis Media's WebBench 5.0 benchmark to generate the HTTP request load for all of the tests. WebBench measures Web server throughput performance by using physical test clients to generate an HTTP based workload against a Web server under test. WebBench reports test results in HTTP requests/second and throughput in bytes per second. Please refer to the Test Methodology section for a detailed description of the test methodology used for each test. We performed two separate runs of each test to verify consistent results and we rebooted the WebBench clients and Web servers before each run.

## SSL connections/second test results

For the SSL connections/second test, we created a custom WebBench test suite that sent HTTP 1.0 SSL requests for a 223 byte static object from the WebBench clients to a virtual IP address configured on the Sun N2000 Series switch. For each request, the Sun N2000 Series switch terminated the TCP connection from the client, negotiated an SSL session with the client, performed SSL decryption, and load balanced the request across a pool of Web servers. The Web server sent its response in clear text to the Sun N2000 Series switch, which encrypted the data and returned the response to the client.

Figure 1 shows the results of each test run as well as their average for the SSL connections/second test.

The Sun N2000 Series switch averaged 12,569 SSL connections/second in our test configuration.

Note that we disabled SSL session reuse in the WebBench test suite, which required the Sun N2000 Series switch to perform a full SSL session negotiation for each incoming request. This increased the amount of SSL processing required by the Sun N2000 Series switch.

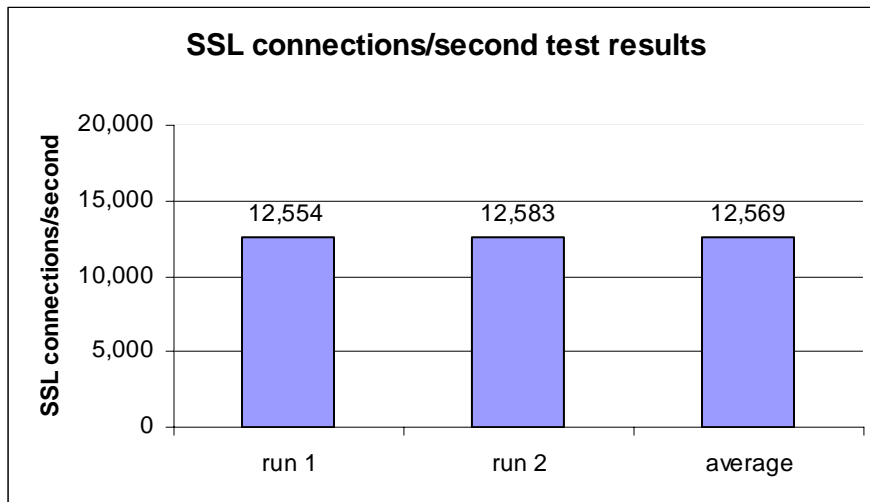


Figure 1: Sun N2000 Series SSL connections/second test results

### SSL bulk cryptographic throughput test results

Figure 2 shows the results for the SSL bulk cryptographic test. We created a custom WebBench test suite that sent HTTP 1.0 SSL requests for a 1MB static object from the WebBench clients to a virtual IP address configured on the Sun N2000 Series switch. For each request, the Sun N2000 Series switch terminated the TCP connection from the client, negotiated an SSL session with the client, performed SSL decryption, and load balanced the request across a pool of Web servers. The Web server sent its response in clear text to the Sun N2000 Series switch, which encrypted the data and returned the response to the client.

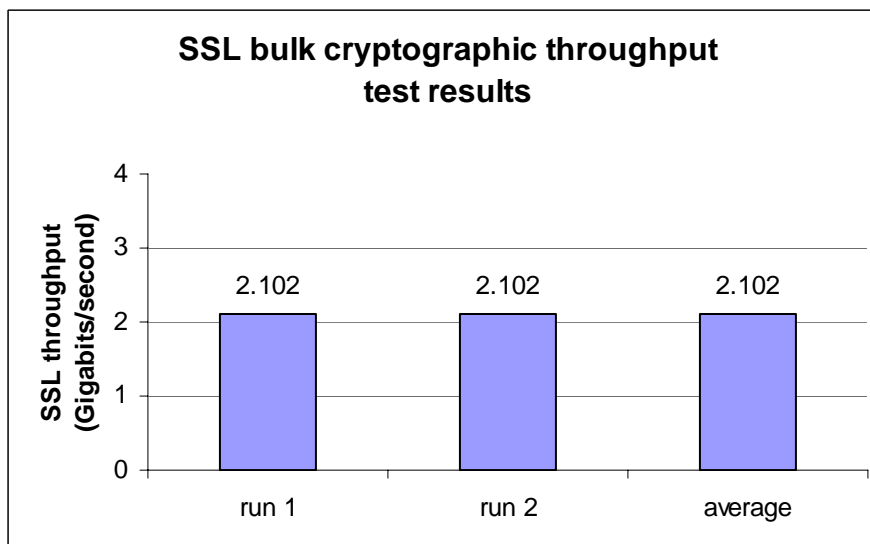


Figure 2: Sun N2000 Series SSL bulk cryptographic throughput test results

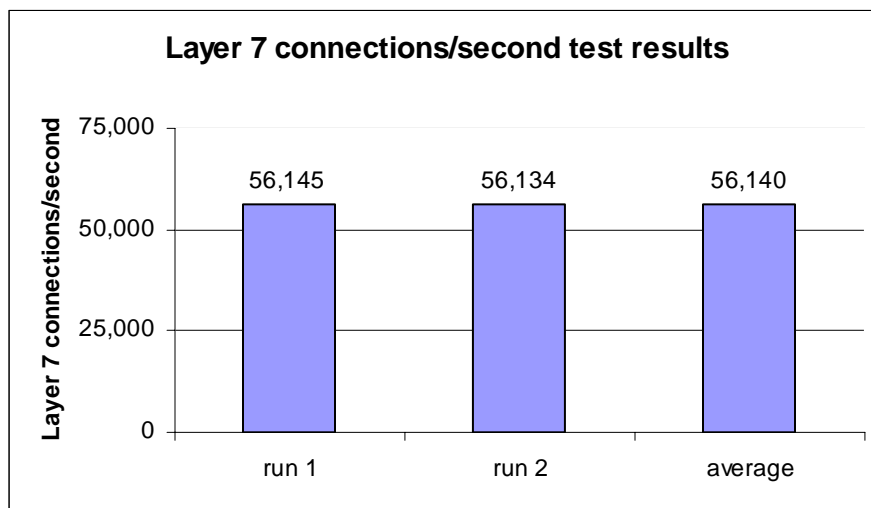
The Sun N2000 Series switch averaged 2.102 Gigabits/second of encrypted SSL throughput during our two test runs. As in the SSL connections/second test, we disabled SSL session reuse on the WebBench clients requiring the switch to negotiate a new SSL session for each request. We checked the utilization on each of the Gigabit Web server connections to the Sun N2000 series switch during the middle of each test run and verified that there was an even throughput distribution and that the utilization numbers matched those reported by the switch.

Note that the throughput results reported by WebBench only reflect the amount of HTTP data encrypted during the test. They do not include the Ethernet, IP, TCP, and HTTP protocol overhead, so the actual number of total throughput bytes moved by the Sun N2000 Series switch is approximately 5% higher due to the protocol overhead.

### ***Layer 7 connections/second test results***

Figure 3 shows the results of each test run as well as their average for the Layer 7 connections/second test. We created a custom WebBench test suite that sent HTTP 1.0 requests for a 223 byte static object from the WebBench clients to a virtual IP address configured on the Sun N2000 Series switch. For each request, the switch terminated the TCP connection from the client, inspected the Layer 7 contents of the request, and load balanced the request based on the Layer 7 content across a pool of Web servers. The Web server sent its response to the Sun N2000 Series switch, which returned the data to the client.

The Sun N2000 Series switch averaged 56,140 Layer 7 connections/second in our test configuration.



**Figure 3: Sun N2000 Series Layer 7 connections/second test results**

### ***Layer 7 throughput test results***

Figure 4 shows the results for the Layer 7 throughput test. We created a custom WebBench test suite that sent HTTP 1.0 requests for a 1MB static object from the WebBench clients to a virtual IP address configured on the Sun N2000 Series switch. For each request, the switch terminated the TCP connection from the client, inspected the Layer 7 contents of the request, and load balanced the request based on the Layer 7 content across a pool of Web servers. The Web server sent its response to the Sun N2000 Series switch, which returned the data to the client.

The Sun N2000 Series switch averaged 2.293 Gigabits/second of Layer 7 throughput during our two test runs. We checked the utilization on each of the Gigabit Web server connections to the Sun N2000 series

switch during the middle of each test run and verified that there was an even throughput distribution and that the utilization numbers matched those reported by the switch.

Note that the throughput results reported by WebBench only reflect the amount of HTTP data transferred during the test. They do not include the Ethernet, IP, TCP, and HTTP protocol overhead, so the actual number of total throughput bytes moved by the Sun N2000 Series switch is approximately 5% higher due to the protocol overhead.

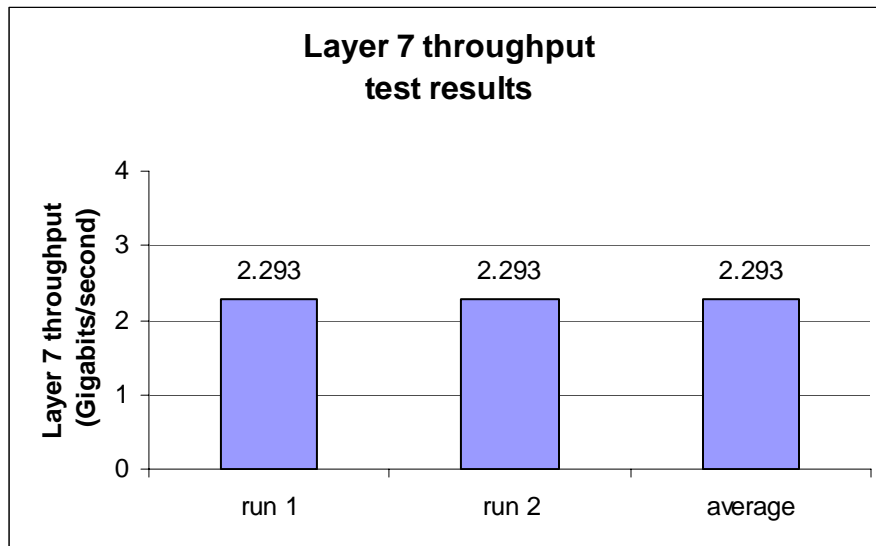


Figure 4: Sun N2000 Series Layer 7 throughput test results

## Testing methodology

Sun Microsystems commissioned VeriTest to measure the SSL and Layer 7 performance of their Sun Secure Application Switch - N2000 Series. Sun supplied a N2120 switch running version V2\_0A47336 firmware and configured with 12 small form factor Gigabit Ethernet ports and one function card. The function card adds support for both hardware-based Layer 4-7 application switching and SSL acceleration.

Sun requested that VeriTest measure the following metrics on the Sun N2000 Series switch:

- Maximum SSL connections/second with no SSL session reuse
- Maximum SSL bulk cryptographic throughput with no SSL session reuse
- Maximum Layer 7 connections/second
- Maximum Layer 7 throughput

For each of the performance tests, the Sun N2000 Series switch was configured as a load balancer and TCP connection termination device positioned between clients generating HTTP requests and Web servers responding to requests. For the SSL tests, the Sun N2000 Series switch terminated the connection request from the client, decrypted the incoming request, load balanced the request across the pool of Web servers, encrypted the response data from the Web server, and returned the response to the client. For the Layer 7 tests, the Sun N2000 Series switch terminated the connection request from the client, inspected the HTTP request contents, load balanced the request across the pool of Web servers, and returned the response to the client.

We used Ziff Davis Media's WebBench 5.0 benchmark to generate the HTTP request load for all of the tests. WebBench measures Web server throughput performance by using physical test clients to generate an HTTP based workload against a Web server under test. As the Web server under test responds to the client

requests, each WebBench client records the number of HTTP requests made and the amount of data moved during the test. Once a test completes, WebBench reports test results in HTTP requests/second and throughput in bytes per second.

The following sections describe the test methodology for each test. Note that we performed two separate runs of each test to verify consistent results and we rebooted the WebBench clients and Web servers before each run. Refer to Appendix A for detailed information on the hardware and software used during the tests.

### ***SSL connections/second test***

For the SSL connections/second test, we created a custom WebBench test suite that sent HTTP 1.0 SSL requests for a 223 byte static object from the WebBench clients to a virtual IP address configured on the Sun N2000 Series switch. For each request, the Sun N2000 Series switch terminated the TCP connection from the client, negotiated an SSL session with the client, performed SSL decryption, and load balanced the request across a pool of Web servers. The Web server sent its response in clear text to the Sun N2000 Series switch, which encrypted the data and returned the response to the client.

The virtual IP address mapped to a function card installed in the Sun N2000 Series switch. The function card performed SSL acceleration processing in hardware. We disabled SSL session reuse in the WebBench test suite, which meant that each HTTP request required the full SSL negotiation and handshaking process. This increased the amount of SSL processing required by the Sun N2000 Series switch.

We used 115 Dell PowerEdge 350 systems running Windows 2000 Professional SP4 as WebBench clients generating the SSL request load. We connected the client systems to four Extreme Networks Summit48 switches via 100Mbps connections. We connected 30 clients to the first three switches and 25 clients to the fourth switch. The network interfaces on the client systems and the ports on the Summit48 switches were set to auto negotiate resulting in 100Mbps full-duplex links. We grouped the Summit48 switches into two pairs, with the switches in each pair connected by a Fiber Gigabit connection. We connected each pair of the Summit48 switches to the Sun N2000 Series switch with a Fiber Gigabit connection resulting in 2Gbps of bandwidth in each direction between the clients and the Sun N2000 Series switch. We configured a single client VLAN on the Sun N2000 Series switch containing the ports connected to the client systems. We set the gateway address on the client systems to the IP address of the client VLAN on the Sun N2000 Series switch.

We used a Gateway E-4600 system running Windows XP Professional SP1 as the WebBench 5.0 controller. We installed an Intel PRO/1000 MF Server Adapter in the controller and connected it to the client VLAN on the Sun N2000 Series switch. We enabled auto negotiation on the Intel adapter and the Sun N2000 Series switch port.

We used 180 Dell PowerEdge 350 systems as the Web servers for the test. Each system ran Windows 2000 Server SP4 with updated patches and Internet Information Server (IIS) 5.0 configured as the Web Server. Using the Microsoft Management Console Internet Services plug-in, we made the following changes to the default IIS 5.0 configuration on each of the Web servers:

- On the WebSite tab, we disabled logging.
- On the Performance tab, we set the IIS performance parameter to 100,000+ hits per day.
- On the Home Directory tab, we set Execute Permissions to None for the document root directory.
- On the Home Directory tab, we disabled the "Index This Resource" property for the document root directory.
- On the Home Directory tab, we set the Application Protection to "Low (IIS Process)" for the document root directory.



Figure 6 shows the port connections on the Sun N2000 Series switch for the SSL connections/second test. All ports on the Sun N2000 Series switch were set to auto negotiate link speed and duplex settings.

Port	Connection
1	Clients 1 – 60 (client VLAN)
2	unused
3	WebBench 5.0 controller (client VLAN)
4	Web Servers 1 – 60 (server VLAN)
5	Clients 61 – 115 (client VLAN)
6	unused
7	unused
8	Web Servers 61 – 120 (server VLAN)
9	unused
10	unused
11	unused
12	Web Servers 121 – 180 (server VLAN)

**Figure 6: Sun N2000 Series port connections for the SSL connections/second test**

An on-site Sun engineer configured the Sun N2000 Series switch for the test. A VeriTest engineer performed the actual test. The Sun N2000 Series switch configuration specified one service group containing all 180 Web servers. The service group used the round robin load balance type to distribute HTTP requests among the Web servers in the group. The configuration included two request policies each containing a single Layer 7 rule. The highest priority request policy matched all objects with the “jpg” suffix. The second request policy (which matched all the WebBench client requests) matched all incoming requests (URI\_PATH matches “\*”). Each of the request policies mapped to the service group. The configuration included two virtual services. The first virtual service had the HTTP service type and the second virtual service had the HTTPS service type and a self-signed certificate. Each virtual service mapped to the two request policies and to the function card in the Sun N2000 Series switch. Each virtual service also contained the virtual IP address accessed by the WebBench clients. Finally, the Sun N2000 Series configuration added the RSA\_EXPORT\_WITH\_RC4\_40\_MD5 cipher to the list of supported SSL ciphers.

Figure 7 lists a summary of the Sun N2000 Series switch configuration for the SSL connections/second test. Refer to Appendix B for a listing of the switch configuration file used for the test.

Switch Setting	Value
LoadBalance->serviceGroup->In Line Health Check	Disabled
LoadBalance->requestPolicy->Optimize Last Response	Enabled
LoadBalance->requestPolicy->First Object Switching	Enabled
Resource->serviceBandwidth->Service Percent	100

**Figure 7: Sun N2000 Series switch configuration changes for the SSL connections/second test**

We created a custom WebBench 5.0 test suite that generated HTTP SSL requests for a 223 byte static object located on each Web server. All of the clients sent their requests to the virtual IP address configured on the Sun N2000 Series switch. The test suite contained one mix that included all 115 WebBench clients in the test. We set the ramp up time and the delay time to 60 seconds. This staggered the start time of the clients for the first 60 seconds of the test. We configured a 10 second ramp down period and an overall test length of 10 minutes. We set the engines per client and the number of threads to 1. We set the thread inactivity timeout to 120 seconds, modified the workload file to specify the URL of the 223 byte file, and configured 100% percent of the requests to use SSL. Finally, we set the min and max values for the SSL reuse session identifiers to 0. This required the clients to renegotiate a new SSL session for each request and increased the amount of load on the Sun N2000 Series switch.

### ***SSL bulk cryptographic throughput test***

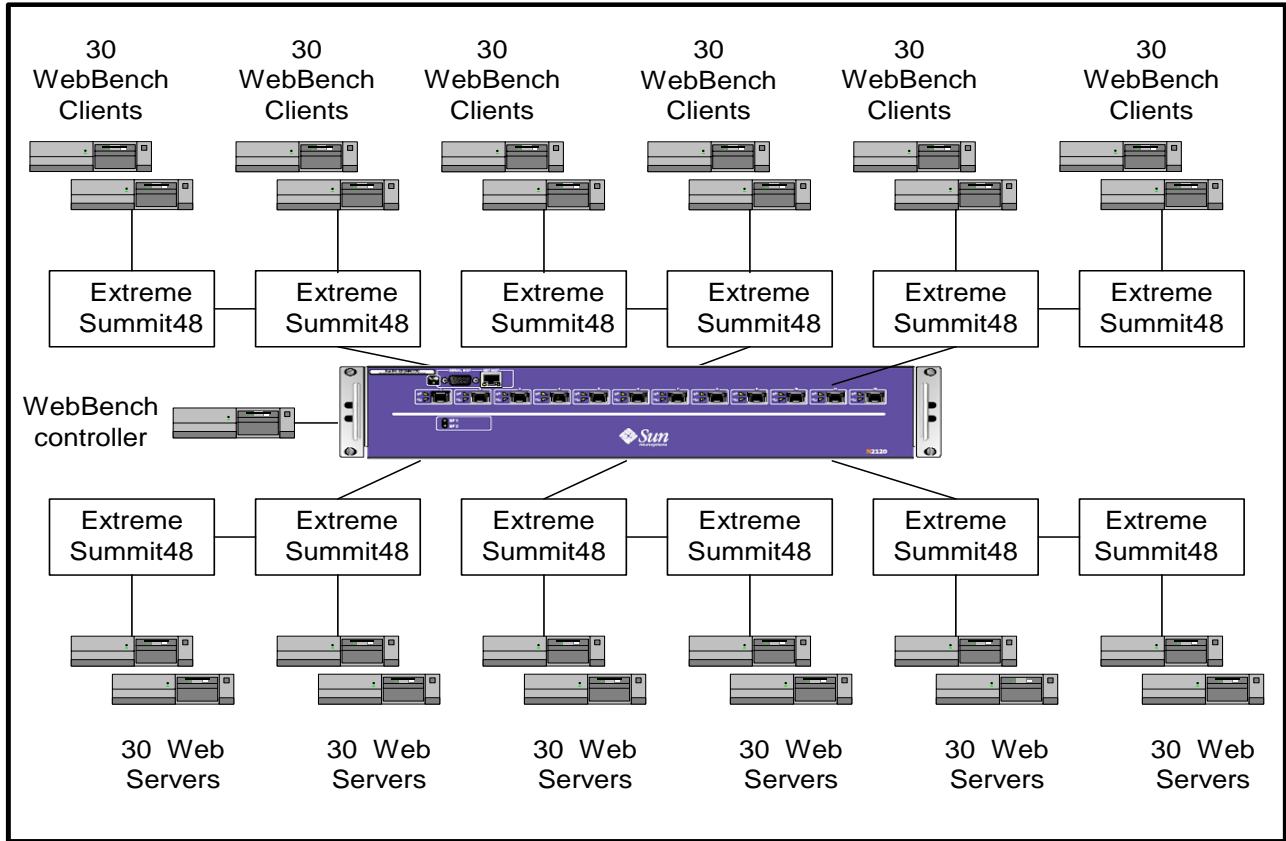
For the SSL bulk cryptographic throughput test, we created a custom WebBench test suite that sent HTTP 1.0 SSL requests for a 1MB static object from the WebBench clients to a virtual IP address configured on the Sun N2000 Series switch. For each request, the Sun N2000 Series switch terminated the TCP connection from the client, negotiated an SSL session with the client, performed SSL decryption, and load balanced the request across a pool of Web servers. The Web server sent its response in clear text to the Sun N2000 Series switch, which encrypted the data and returned the response to the client.

We used 180 Dell PowerEdge 350 systems as WebBench clients generating the HTTP request load. The client systems ran a mixture of Windows 2000 Professional SP4 (120 systems) and Windows XP Professional SP1 (60 systems). We connected the client systems to six Extreme Networks Summit48 switches via 100Mbps connections with 30 clients connected to each switch. The network interfaces on the client systems and the ports on the Summit48 switches were set to auto negotiate resulting in 100Mbps full-duplex links. We divided the Summit48 switches into three groups. Each group contained two switches and 60 clients. The switches within each group were connected by Fiber Gigabit connections. We connected each group of Summit48 switches to the Sun N2000 Series switch via a Fiber Gigabit connection resulting in 3Gbps of bandwidth in each direction between the clients and the Sun N2000 Series switch. We configured a single client VLAN on the Sun N2000 Series switch containing the ports connected to the client systems. We set the gateway address on the client systems to the IP address of the client VLAN on the Sun N2000 Series switch.

We used the same Gateway E-4600 system used in the SSL connections/second test as the WebBench 5.0 controller. We connected it to the client VLAN on the Sun N2000 Series switch. We enabled auto negotiation on the controller and the Sun N2000 Series switch port.

We used the same Web server configuration described in SSL connection/second test for the SSL bulk cryptographic throughput test. The only change was to copy a 1M byte static file to the document root on each Web server.

Figure 8 illustrates the test bed configuration used for the SSL bulk cryptographic throughput test.



**Figure 8: SSL bulk cryptographic throughput test bed**

Figure 9 shows the port connections on the Sun N2000 Series switch for the SSL bulk cryptographic throughput test.

Port	Connection
1	Clients 1 – 60 (client VLAN)
2	unused
3	WebBench 5.0 controller (client VLAN)
4	Web Servers 1 – 60 (server VLAN)
5	Clients 61 – 120 (client VLAN)
6	unused
7	unused
8	Web Servers 61 – 120 (server VLAN)
9	Clients 121 – 180 (client VLAN)
10	unused
11	unused
12	Web Servers 121 – 180 (server VLAN)

**Figure 9: Sun N2000 Series port connections for the SSL bulk cryptographic test**

Figure 10 lists a summary of the Sun N2000 Series switch configuration for the SSL bulk cryptographic throughput test. Refer to Appendix B for a listing of the switch configuration file used for the test.

Switch Setting	Value
LoadBalance->realService->Advanced->Receive Window Size	65535
LoadBalance->realService->Advanced->SMM Stream Limit	4xRcvWindow
LoadBalance->virtualService->Advanced->Receive Window Size (HTTP)	65535
LoadBalance->virtualService->Advanced->SMM Stream Limit (HTTP)	4xRcvWindow
LoadBalance->virtualService->Advanced->Receive Window Size (HTTPS)	65535
LoadBalance->virtualService->Advanced->SMM Stream Limit (HTTPS)	4xRcvWindow
LoadBalance->serviceGroup->In Line Health Check	Disabled
LoadBalance->requestPolicy->Optimize Last Response	Enabled
LoadBalance->requestPolicy->First Object Switching	Enabled
System->Switch Services->tideRunner->initKeys->SMM Page Size	8
Resource->serviceBandwidth->Service Percent	100

**Figure 10: Sun N2000 Series switch configuration changes for the SSL bulk cryptographic throughput test**

We created a custom WebBench 5.0 test suite that generated HTTP SSL requests for a 1M byte static object located on each Web server. All of the clients sent their requests to the virtual IP address configured on the Sun N2000 Series switch. The test suite contained one mix that included all 180 WebBench clients in the test. We set the ramp up time and the delay time to 60 seconds. This staggered the start time of the clients for the first 60 seconds of the test. We configured a 10 second ramp down period and an overall test length of 10 minutes. We set the engines per client to 5 and the number of threads to 1. We set the thread inactivity timeout to 120 seconds, modified the workload file to specify the URL of the 1M byte file, and configured 100% percent of the requests to use SSL. Finally, we set the min and max values for the SSL reuse session identifiers to 0. This required the clients to renegotiate a new SSL session for each request and increased the amount of load on the Sun N2000 Series switch.

### ***Layer 7 throughput test***

For the Layer 7 throughput test, we created a custom WebBench test suite that sent HTTP 1.0 requests for a 1M byte static object from the WebBench clients to a virtual IP address configured on the Sun N2000 Series switch. For each request, the switch terminated the TCP connection from the client, inspected the Layer 7 contents of the request, and load balanced the request based on the Layer 7 content across a pool of Web servers. The Web server sent its response to the Sun N2000 Series switch, which returned the data to the client.

The virtual IP address mapped to a function card installed in the Sun N2000 Series switch. The function card performed Layer 7 packet inspection in hardware.

We used the same test bed configuration and Sun N2000 Series switch settings defined in the SSL bulk cryptographic throughput test for the Layer 7 throughput test.

We created a custom WebBench 5.0 test suite that generated HTTP 1.0 requests for a 1M byte static object located on each Web server. All of the clients sent their requests to the virtual IP address configured on the Sun N2000 Series switch. The test suite contained one mix that included all 180 WebBench clients in the test. We set the ramp up time and the delay time to 60 seconds. This staggered the start time of the clients for the first 60 seconds of the test. We configured a 10 second ramp down period and an overall test length of 10 minutes. We set the engines per client to 5 and the number of threads to 1. We set the thread inactivity timeout to 120 seconds and modified the workload file to specify the URL of the 1M byte file. We disabled SSL negotiation so that all of the HTTP 1.0 transfers occurred in clear text.

## ***Layer 7 connections/second test***

For the Layer 7 connections/second test, we created a custom WebBench test suite that sent HTTP 1.0 requests for a 223 byte static object from the WebBench clients to a virtual IP address configured on the Sun N2000 Series switch. For each request, the switch terminated the TCP connection from the client, inspected the Layer 7 contents of the request, and load balanced the request based on the Layer 7 content across a pool of Web servers. The Web server sent its response to the Sun N2000 Series switch, which returned the data to the client.

We used the same test bed configuration defined in the SSL bulk cryptographic throughput test for the Layer 7 connections/second test. We used the same Sun N2000 Series switch settings defined in the SSL connections/second test for the Layer 7 connections/second test.

We created a custom WebBench 5.0 test suite that generated HTTP 1.0 requests for a 223 byte static object located on each Web server. All of the clients sent their requests to the virtual IP address configured on the Sun N2000 Series switch. The test suite contained one mix that included all 180 WebBench clients in the test. We set the ramp up time and the delay time to 60 seconds. This staggered the start time of the clients for the first 60 seconds of the test. We configured a 10 second ramp down period and an overall test length of 10 minutes. We set the engines per client to 1 and the number of threads to 10. We set the thread inactivity timeout to 120 seconds and we modified the workload file to specify the URL of the 223 byte file. Like the Layer 7 throughput test, we configured WebBench to use HTTP 1.0 connections with clear-text data transfers. SSL negotiation was not performed on any of the requests.

## Appendix

### A. Test Hardware and Software

Sun N2000 Series switch	
Model	N2120
Software version	V2_0A47336
Number of Gigabit ports	12 small form factor fiber Gigabit ports
Number of function cards	1

Figure 11: Sun N2000 Series switch

WebBench clients (180 identical systems)	
Machine Type	Dell PowerEdge 350
BIOS	TR440BXA.86B.0034.P12
Processor	850MHz Pentium III
L2 Cache	256KB
Expansion Bus	PCI
Memory	256MB
Disk(s)	10GB Maxtor ST010H1, Microsoft driver version: 5.0.2183.1 (W2K), Microsoft driver version 5.1.2535.0 (XP)
Network Adapter(s)	2 embedded Intel PRO 100+, Intel driver version 8.0.11.0 (W2K), Microsoft driver version 5.41.22.0 (XP)
OS	Windows 2000 Professional SP4 (120 systems) Windows XP Professional SP1 (60 systems)

Figure 12: WebBench client systems

Web servers (180 identical systems)	
Machine Type	Dell PowerEdge 350
BIOS	TR440BXA.86B.0034.P12
Processor	850MHz Pentium III
L2 Cache	256KB
Expansion Bus	PCI
Memory	256MB
Disk(s)	10GB Maxtor ST010H1, Microsoft driver version 5.0.2183.1
Network Adapter(s)	2 embedded Intel PRO 100+, Intel driver version 8.0.11.0
OS	Windows 2000 Server SP4

Figure 13: Web server systems

WebBench Controller	
Machine Type	Gateway E-4600
BIOS	GB85010A.15A.0046.P13.0108201551 8/20/2001
Processor	1.3GHz Pentium 4
L2 Cache	256KB
Expansion Bus	PCI
Memory	256MB
Disk(s)	WDC WD400BB-53AUA1, Microsoft driver version 5.1.2535.0
Network Adapter(s)	Intel PRO/100+, Intel driver version 7.1.8.0 Intel PRO/1000 MF Server Adapter, Intel driver version 8.0.57.0
OS	Windows XP Professional
OS Updates	Service Pack 1

Figure 14: WebBench controller system

## B. Sun N2000 Series switch configuration files

### B.1 SSL and Layer 7 connections/second test configuration file

```
#####
# System Name: Sun Application Switch
# Date : Mon Oct 18 17:15:18 2004
# Serial No : 02042700099
#####
commandModeEntry on

config
#
# Event configuration and statistics
#
event
#
# Profile rules for event filters
#
filterProfile defaultFile {default filter for saving to file}
filterProfile defaultFile
#
# Profiles to cause event filtering
#
rule 130 drop logLevel warning
rule 200 send true
exit;
filterProfile defaultLog {default log filter}
filterProfile defaultLog
#
# Profiles to cause event filtering
#
rule 90 drop logLevel debug
rule 100 send true
exit;
filterProfile defaultSyslog {default syslog filter}
filterProfile defaultSyslog
#
# Profiles to cause event filtering
#
rule 90 drop logLevel debug
rule 100 send true
exit;
filterProfile defaultTrapd {default trapd filter}
filterProfile defaultTrapd
```

```

#
# Profiles to cause event filtering
#
rule 90 drop logLevel warning
rule 100 send true
exit;
exit;
#
# NMON state and status
#
nmon
exit;
#
# Port configuration
#
port eth.1.1 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan clients
port eth.1.1
exit;
port eth.1.2 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex
port eth.1.2
exit;
port eth.1.3 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan clients
port eth.1.3
exit;
port eth.1.4 phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan servers
port eth.1.4
exit;
port eth.1.5 phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan clients
port eth.1.5
exit;
port eth.1.6 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex
port eth.1.6
exit;
port eth.1.7 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex
port eth.1.7
exit;
port eth.1.8 phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan servers
port eth.1.8
exit;
port eth.1.9 phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan clients
port eth.1.9
exit;
port eth.1.10 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex
port eth.1.10
exit;
port eth.1.11 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex
port eth.1.11
exit;
port eth.1.12 phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan servers
port eth.1.12
exit;
#
# vSwitch configuration
#
vSwitch system {System vSwitch}
vSwitch system
#
# Port Bandwidth configuration
#
resource portBandwidth eth.1.1 100 100 65534 65535
resource portBandwidth eth.1.3 100 100 65534 65535
resource portBandwidth eth.1.5 100 100 65534 65535
resource portBandwidth eth.1.9 100 100 65534 65535
#
# vRouter configuration
#
vRouter management {System Management vRouter}
vRouter management
#
# Display IP layer configuration
#

```

```

ip forwarding disabled
ip
#
# IP Interfaces
#
interface ethMgmt.1
interface ethMgmt.1 linkUpDownTrap disabled
#
# IP Interface Address
#
address ethMgmt.1 10.0.0.1 255.255.255.0
#
# ICMP configuration
#
icmp true true false true
#
# TCP Connections
#
tcp connections 0.0.0.0 23 0.0.0.0 0 listen
tcp connections 0.0.0.0 80 0.0.0.0 0 listen
tcp connections 10.0.0.1 80 10.0.0.10 1179 established
tcp connections 10.0.0.1 80 10.0.0.10 1180 established
exit;
#
# Interfaces
#
interfaces sock.system:management linkUpDownTrap disabled
interfaces sock.system:management/ip.system:management linkUpDownTrap disabled
interfaces ip.system:management linkUpDownTrap disabled
interfaces ip.system:management/ethMgmt.1 linkUpDownTrap disabled
interfaces ethMgmt.1 linkUpDownTrap disabled
interfaces loopback linkUpDownTrap disabled
#
# VRRP configuration
#
vrrp
exit;
exit;
vRouter shared {Shared vRouter}
vRouter shared
#
# VLAN configuration parameters
#
vlan clients 10
vlan clients linkUpDownTrap disabled
vlan clients
#
# VLAN interface configuration
#
interface eth.1.1
interface eth.1.1 linkUpDownTrap disabled
interface eth.1.3
interface eth.1.3 linkUpDownTrap disabled
interface eth.1.5
interface eth.1.5 linkUpDownTrap disabled
interface eth.1.9
interface eth.1.9 linkUpDownTrap disabled
#
# VLAN STP Interface Configuration
#
interface spanningTree eth.1.1 pathCost 4
interface spanningTree eth.1.3 pathCost 4
interface spanningTree eth.1.5 pathCost 4
interface spanningTree eth.1.9 pathCost 4
exit;
#
# Display IP layer configuration
#
ip forwarding enabled
ip
#

```

```

# IP Interfaces
#
interface vlan.clients
interface vlan.clients linkUpDownTrap disabled
#
# IP Interface Address
#
address vlan.clients 10.0.0.254 255.255.255.0
#
# ICMP configuration
#
icmp true true false true
exit;
#
# Interfaces
#
interfaces sock.system:shared linkUpDownTrap disabled
interfaces sock.system:shared/ip.system:shared linkUpDownTrap disabled
interfaces ip.system:shared linkUpDownTrap disabled
interfaces ip.system:shared/vlan.clients linkUpDownTrap disabled
interfaces vlan.clients linkUpDownTrap disabled
interfaces vlan.clients/eth.1.1 linkUpDownTrap disabled
interfaces vlan.clients/eth.1.5 linkUpDownTrap disabled
interfaces vlan.clients/eth.1.9 linkUpDownTrap disabled
interfaces vlan.clients/eth.1.3 linkUpDownTrap disabled
interfaces loopback linkUpDownTrap disabled
#
# VRRP configuration
#
vrrp
exit;
exit;
vSwitch veritest
#
# Export a private key or certificate from the system
#
ckm import paste veritest certificate internalCkm
-----BEGIN CERTIFICATE-----
MIICDCCAXGgAwIBAgIBATANBgkqhkiG9w0BAQUFADAbMRkwFwYDVQQDExB3d3cu
dmVyaXRlc3QuY29tMB4XDTA0MTAxODEwMzlyM1oXDTA1MTAxODEwMzlyM1owGzEZ
MBcGA1UEAxMQd3d3LnZlcmI0ZXN0LmNvbTcBnzANBgkqhkiG9w0BAQEFAAOBjQAw
gYkCgYEA3ndjif6c79lsmsofz4lV8dEuns/09dtmrAnWrDIIzh8cUfc2RRpY82YK/
bQEV3LswF5f77jECoyWFAk4l9d5QKxgl+fgvz0s0hv6LAXl6YY1/UZ2TtNVK1Bm
RXtaypwbuz+ReJWQr5PaPD17Uybz0opifKd7VvJNtLz6bU7a48CAwEAANcMFow
GwYDVR0RBBQwEolQd3d3LnZlcmI0ZXN0LmNvbTAPBgNVHRMBAf8EBTADAQH/MASG
A1UdDwQEAWBtjAdBgNVHQ4EFgQU+F+TquW0AGqCXM2kCHE6oWx8cPMwDQYJKoZI
hvcNAQEFBQADgYEAPlsvB64aSX53UUXTSxuKBo8YuPKN9xb4iOXRHSPJ/i2b2Jq
AVlv7ldpTfHzPYS6appt2Hue9GLxpWobSCarOLUUS5Bij4B7CbuNqrKAXMDhFoTO
wtcO9awwwr/cNKUiyRwpEyF+C6BpOMuPfp0zv1iLWyzqZDs4DBxvN545l/Fk=
-----END CERTIFICATE-----
(ctrl-z)
#
# Host configuration
#
loadBalance host host_1 192.168.1.1 vRouter veritest:default
loadBalance host host_2 192.168.1.2 vRouter veritest:default
loadBalance host host_3 192.168.1.3 vRouter veritest:default

<repetitive entries for hosts 4 – 177 removed>

loadBalance host host_178 192.168.1.178 vRouter veritest:default
loadBalance host host_179 192.168.1.179 vRouter veritest:default
loadBalance host host_180 192.168.1.180 vRouter veritest:default
#
# Expressions used to classify the application data stream
#
loadBalance objectRule default {URI_PATH matches ""*}
loadBalance objectRule jpg {URI_SUFFIX eq ".jpg"}
#
# Real service parameters

```

```

#
loadBalance realService rs_clear_1 host_1 certType Literal
loadBalance realService rs_clear_1
exit; exit;
loadBalance realService rs_clear_2 host_2 certType Literal
loadBalance realService rs_clear_2
exit; exit;
loadBalance realService rs_clear_3 host_3 certType Literal
loadBalance realService rs_clear_3
exit; exit;

<repetitive entries for realServices 4 – 177 removed>

loadBalance realService rs_clear_178 host_178 certType Literal
loadBalance realService rs_clear_178
exit; exit;
loadBalance realService rs_clear_179 host_179 certType Literal
loadBalance realService rs_clear_179
exit; exit;
loadBalance realService rs_clear_180 host_180 certType Literal
loadBalance realService rs_clear_180
exit; exit;
loadBalance realService rs_secure_1 host_1 port 443 certType Literal
loadBalance realService rs_secure_1
exit; exit;
loadBalance realService rs_secure_2 host_2 port 443 certType Literal
loadBalance realService rs_secure_2
exit; exit;
loadBalance realService rs_secure_3 host_3 port 443 certType Literal
loadBalance realService rs_secure_3
exit; exit;

<repetitive entries for realServices 4 – 177 removed>

loadBalance realService rs_secure_178 host_178 port 443 certType Literal
loadBalance realService rs_secure_178
exit; exit;
loadBalance realService rs_secure_179 host_179 port 443 certType Literal
loadBalance realService rs_secure_179
exit; exit;
loadBalance realService rs_secure_180 host_180 port 443 certType Literal
loadBalance realService rs_secure_180
exit; exit;
#
# Request Policies
#
loadBalance requestPolicy QP_DEFAULT forward default sg_clear 10 optimizeLastResponse enabled firstObjectSwitching enabled
loadBalance requestPolicy QP_DEFAULT
exit; exit;
loadBalance requestPolicy QP_JPG forward jpg sg_clear optimizeLastResponse enabled firstObjectSwitching enabled
loadBalance requestPolicy QP_JPG
exit; exit;
#
# Service group configuration
#
loadBalance serviceGroup sg_clear roundRobin \
{rs_clear_1; rs_clear_10; rs_clear_100; rs_clear_101; rs_clear_102; rs_clear_103; rs_clear_104; rs_clear_105; rs_clear_106;
rs_clear_107; rs_clear_108; rs_clear_109; rs_clear_11; rs_clear_110; rs_clear_111; rs_clear_112; rs_clear_113; rs_clear_114;
rs_clear_115; rs_clear_116; rs_clear_117; rs_clear_118; rs_clear_119; rs_clear_12; rs_clear_120; rs_clear_121; rs_clear_122;
rs_clear_123; rs_clear_124; rs_clear_125; rs_clear_126; rs_clear_127; rs_clear_128; rs_clear_129; rs_clear_13; rs_clear_130;
rs_clear_131; rs_clear_132; rs_clear_133; rs_clear_134; rs_clear_135; rs_clear_136; rs_clear_137; rs_clear_138; rs_clear_139;
rs_clear_14; rs_clear_140; rs_clear_141; rs_clear_142; rs_clear_143; rs_clear_144; rs_clear_145; rs_clear_146; rs_clear_147;
rs_clear_148; rs_clear_149; rs_clear_15; rs_clear_150; rs_clear_151; rs_clear_152; rs_clear_153; rs_clear_154; rs_clear_155;
rs_clear_156; rs_clear_157; rs_clear_158; rs_clear_159; rs_clear_16; rs_clear_160; rs_clear_161; rs_clear_162; rs_clear_163;
rs_clear_164; rs_clear_165; rs_clear_166; rs_clear_167; rs_clear_168; rs_clear_169; rs_clear_17; rs_clear_170; rs_clear_171;
rs_clear_172; rs_clear_173; rs_clear_174; rs_clear_175; rs_clear_176; rs_clear_177; rs_clear_178; rs_clear_179; rs_clear_18;
rs_clear_180; rs_clear_19; rs_clear_2; rs_clear_20; rs_clear_21; rs_clear_22; rs_clear_23; rs_clear_24; rs_clear_25; rs_clear_26;
rs_clear_27; rs_clear_28; rs_clear_29; rs_clear_3; rs_clear_30; rs_clear_31; rs_clear_32; rs_clear_33; rs_clear_34; rs_clear_35;
rs_clear_36; rs_clear_37; rs_clear_38; rs_clear_39; rs_clear_4; rs_clear_40; rs_clear_41; rs_clear_42; rs_clear_43; rs_clear_44;
rs_clear_45; rs_clear_46; rs_clear_47; rs_clear_48; rs_clear_49; rs_clear_5; rs_clear_50; rs_clear_51; rs_clear_52; rs_clear_53;
rs_clear_54; rs_clear_55; rs_clear_56; rs_clear_57; rs_clear_58; rs_clear_59; rs_clear_6; rs_clear_60; rs_clear_61; rs_clear_62;

```

```

rs_clear_63; rs_clear_64; rs_clear_65; rs_clear_66; rs_clear_67; rs_clear_68; rs_clear_69; rs_clear_7; rs_clear_70; rs_clear_71;
rs_clear_72; rs_clear_73; rs_clear_74; rs_clear_75; rs_clear_76; rs_clear_77; rs_clear_78; rs_clear_79; rs_clear_8; rs_clear_80;
rs_clear_81; rs_clear_82; rs_clear_83; rs_clear_84; rs_clear_85; rs_clear_86; rs_clear_87; rs_clear_88; rs_clear_89; rs_clear_9;
rs_clear_90; rs_clear_91; rs_clear_92; rs_clear_93; rs_clear_94; rs_clear_95; rs_clear_96; rs_clear_97; rs_clear_98; rs_clear_99;
rs_clear_0} \
  inlineHealthCheck disabled
  loadBalance serviceGroup sg_clear
  exit; exit;
#
# Virtual Service Group Configuration
#
loadBalance vsGroup default {veritest-clear; veritest_secure}
#
# Virtual Service configuration
#
loadBalance virtualService veritest-clear HTTP 10.0.0.250 {QP_DEFAULT; QP_JPG}
loadBalance virtualService veritest-clear
#
# Virtual service advanced settings
#
  advanced smmStreamLimit 8xRcvWnd initParseWithData true
exit; exit;
loadBalance virtualService veritest_secure HTTPS 10.0.0.250 {QP_DEFAULT; QP_JPG} ckmKeyName veritest sslCiphers \
{RSA_EXPORT_WITH_RC4_40_MD5; RSA_WITH_RC4_128_MD5; RSA_WITH_RC4_128_SHA;
RSA_EXPORT_WITH_DES40_CBC_SHA; RSA_WITH_DES_CBC_SHA; RSA_WITH_3DES_EDE_CBC_SHA;
RSA_WITH_AES_128_CBC_SHA; RSA_EXPORT1024_WITH_DES_CBC_SHA; RSA_EXPORT1024_WITH_RC4_56_SHA}
loadBalance virtualService veritest_secure
#
# Virtual service advanced settings
#
  advanced initParseWithData true
exit; exit;
#
# Port Bandwidth configuration
#
resource portBandwidth eth.1.1 100 100 65534 65535
resource portBandwidth eth.1.3 100 100 65534 65535
resource portBandwidth eth.1.4 100 100 65534 65535
resource portBandwidth eth.1.5 100 100 65534 65535
resource portBandwidth eth.1.8 100 100 65534 65535
resource portBandwidth eth.1.9 100 100 65534 65535
resource portBandwidth eth.1.12 100 100 65534 65535
#
# Service Engine Bandwidth Configuration
#
resource serviceBandwidth functionCard1 100 100
#
# vRouter configuration
#
vRouter default {Default vRouter}
vRouter default
#
# VLAN configuration parameters
#
vlan servers 20
vlan servers linkUpDownTrap disabled
vlan servers
#
# VLAN interface configuration
#
interface eth.1.4
interface eth.1.4 linkUpDownTrap disabled
interface eth.1.8
interface eth.1.8 linkUpDownTrap disabled
interface eth.1.12
interface eth.1.12 linkUpDownTrap disabled
#
# VLAN STP Interface Configuration
#
interface spanningTree eth.1.4 pathCost 4
interface spanningTree eth.1.8 pathCost 4

```

```

    interface spanningTree eth.1.12 pathCost 4
exit;
#
# Display IP layer configuration
#
ip forwarding enabled
ip
#
# IP Interfaces
#
interface vlan.servers
interface vlan.servers linkUpDownTrap disabled

#
# IP Interface Address
#
address vlan.servers 192.168.1.254 255.255.255.0

#
# ICMP configuration
#
icmp true true false true
exit;
#
# Interfaces
#
interfaces sock.veritest:default linkUpDownTrap disabled
interfaces sock.veritest:default/ip.veritest:default linkUpDownTrap disabled
interfaces ip.veritest:default linkUpDownTrap disabled
interfaces ip.veritest:default/vlan.servers linkUpDownTrap disabled
interfaces vlan.servers linkUpDownTrap disabled
interfaces vlan.servers/eth.1.4 linkUpDownTrap disabled
interfaces vlan.servers/eth.1.8 linkUpDownTrap disabled
interfaces vlan.servers/eth.1.12 linkUpDownTrap disabled
interfaces loopback linkUpDownTrap disabled
#
# VRRP configuration
#
vrrp
exit;
exit;
#
# HTTP configuration and status
#
switchServices httpd enabled
switchServices httpd
exit; exit;
#
# Global NTP configuration parameters
#
switchServices ntp
#
# Global NTP configuration parameters (advanced)
#
advanced
exit;
exit; exit;
#
# SNMP configuration
#
switchServices snmp
#
# SNMP engine configuration
#
systemInfo {} {Sun Application Switch} {}
exit; exit;
#
# Software key
#
switchServices software key {}

```

```

#
# SSHd configuration and operation
#
switchServices sshd confEncryption {des3Cbc; blowfishCbc; des} confHmac \
  {md5; sha1; md5b96; sha1b96} userAuthentication {publicKey; password}
switchServices sshd
#
# SSHd configuration and operation (advanced)
#
  advanced
  exit;
exit; exit;
#
# Telnetd configuration and current status
#
switchServices telnetd enabled
switchServices telnetd
exit; exit;
#
# TFTPd configuration and session statistics
#
switchServices tftpd
exit; exit;
#
# TideRunner Configuration
#
switchServices tideRunner initkeys functionCard1 20000 statPollPeriod 5 smmPageSize 2 dleMaxHdrLen 8192
#
# Configuration and status for the trap process
#
switchServices trap
exit; exit;

```

## ***B.2 SSL bulk cryptographic and Layer 7 throughput test configuration file***

```

#####
# System Name: Sun Application Switch
# Date   : Tue Oct 19 11:17:08 2004
# Serial No : 02042700099
#####
commandModeEntry on

config
#
# Event configuration and statistics
#
event
#
# Profile rules for event filters
#
filterProfile defaultFile {default filter for saving to file}
filterProfile defaultFile
#
# Profiles to cause event filtering
#
rule 130 drop logLevel warning
rule 200 send true
exit;
filterProfile defaultLog {default log filter}
filterProfile defaultLog
#
# Profiles to cause event filtering
#
rule 90 drop logLevel debug
rule 100 send true
exit;
filterProfile defaultSyslog {default syslog filter}
filterProfile defaultSyslog
#

```

```

    # Profiles to cause event filtering
    #
    rule 90 drop logLevel debug
    rule 100 send true
exit;
filterProfile defaultTrapd {default trapd filter}
filterProfile defaultTrapd
#
# Profiles to cause event filtering
#
    rule 90 drop logLevel warning
    rule 100 send true
exit;
exit;
#
# NMON state and status
#
nmon
exit;
#
# Port configuration
#
port eth.1.1 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan clients
port eth.1.1
exit;
port eth.1.2 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex
port eth.1.2
exit;
port eth.1.3 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan clients
port eth.1.3
exit;
port eth.1.4 phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan servers
port eth.1.4
exit;
port eth.1.5 phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan clients
port eth.1.5
exit;
port eth.1.6 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex
port eth.1.6
exit;
port eth.1.7 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex
port eth.1.7
exit;
port eth.1.8 phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan servers
port eth.1.8
exit;
port eth.1.9 phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan clients
port eth.1.9
exit;
port eth.1.10 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex
port eth.1.10
exit;
port eth.1.11 phySpeed 1000M phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex
port eth.1.11
exit;
port eth.1.12 phyDuplex fullDuplex advSpeed 1000M advDuplex fullDuplex defVlan servers
port eth.1.12
exit;
#
# vSwitch configuration
#
vSwitch system {System vSwitch}
vSwitch system
#
# Port Bandwidth configuration
#
resource portBandwidth eth.1.1 100 100 65534 65535
resource portBandwidth eth.1.3 100 100 65534 65535
resource portBandwidth eth.1.5 100 100 65534 65535
resource portBandwidth eth.1.9 100 100 65534 65535

```

```

#
# vRouter configuration
#
vRouter management {System Management vRouter}
vRouter management
#
# Display IP layer configuration
#
ip forwarding disabled
ip
#
# IP Interfaces
#
interface ethMgmt.1
interface ethMgmt.1 linkUpDownTrap disabled
#
# IP Interface Address
#
address ethMgmt.1 10.0.0.1 255.255.255.0
#
# ICMP configuration
#
icmp true true false true
#
# TCP Connections
#
tcp connections 0.0.0.0 23 0.0.0.0 0 listen
tcp connections 0.0.0.0 80 0.0.0.0 0 listen
tcp connections 10.0.0.1 80 10.0.0.2 1074 established
tcp connections 10.0.0.1 80 10.0.0.2 1075 established
exit;
#
# Interfaces
#
interfaces sock.system:management linkUpDownTrap disabled
interfaces sock.system:management/ip.system:management linkUpDownTrap disabled
interfaces ip.system:management linkUpDownTrap disabled
interfaces ip.system:management/ethMgmt.1 linkUpDownTrap disabled
interfaces ethMgmt.1 linkUpDownTrap disabled
interfaces loopback linkUpDownTrap disabled
#
# VRRP configuration
#
vrrp
exit;
exit;
vRouter shared {Shared vRouter}
vRouter shared
#
# VLAN configuration parameters
#
vlan clients 10
vlan clients linkUpDownTrap disabled
vlan clients
#
# VLAN interface configuration
#
interface eth.1.1
interface eth.1.1 linkUpDownTrap disabled
interface eth.1.3
interface eth.1.3 linkUpDownTrap disabled
interface eth.1.5
interface eth.1.5 linkUpDownTrap disabled
interface eth.1.9
interface eth.1.9 linkUpDownTrap disabled
#
# VLAN STP Interface Configuration
#
interface spanningTree eth.1.1 pathCost 4
interface spanningTree eth.1.3 pathCost 4
interface spanningTree eth.1.5 pathCost 4

```

```

    interface spanningTree eth.1.9 pathCost 4
exit;
#
# Display IP layer configuration
#
ip forwarding enabled
ip
#
# IP Interfaces
#
interface vlan.clients
interface vlan.clients linkUpDownTrap disabled
#
# IP Interface Address
#
address vlan.clients 10.0.0.254 255.255.255.0
#
# ICMP configuration
#
icmp true true false true
exit;
#
# Interfaces
#
interfaces sock.system:shared linkUpDownTrap disabled
interfaces sock.system:shared/ip.system:shared linkUpDownTrap disabled
interfaces ip.system:shared linkUpDownTrap disabled
interfaces ip.system:shared/vlan.clients linkUpDownTrap disabled
interfaces vlan.clients linkUpDownTrap disabled
interfaces vlan.clients/eth.1.1 linkUpDownTrap disabled
interfaces vlan.clients/eth.1.5 linkUpDownTrap disabled
interfaces vlan.clients/eth.1.9 linkUpDownTrap disabled
interfaces vlan.clients/eth.1.3 linkUpDownTrap disabled
interfaces loopback linkUpDownTrap disabled
#
# VRRP configuration
#
vrrp
exit;
exit;
exit;
vSwitch veritest
#
# Export a private key or certificate from the system
#
ckm import paste veritest certificate internalCkm
-----BEGIN CERTIFICATE-----
MIICDCCAXGgAwIBAgIBATANBgkqhkiG9w0BAQUFADAbMRkwFwYDVQQDExB3d3cu
dmVyaXRlc3QuY29tMB4XDTA0MTAxODEwMzlyM1oXDTA1MTAxODEwMzlyM1owGzEZ
MBCGA1UEAxMQd3d3LnZlcmI0ZXN0LmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAw
gYkCgYEA3ndjf6c79lsmsofz4lV8dEuns/09dtmrAnWrDIIzh8cUfc2RRpY82YK/
bQEV3LswF5f77jECoyWFAk4l9d5QKxgl+fgvz0s0hv6lLaxl6YY1/UZ2TtNVK1Bm
RXtaypwbuRz+ReJWQR5PaPD17Uybz0opifKd7VvJNtLz6bU7a48CAwEAAANcMFow
GwYDVR0RBQQwEolQd3d3LnZlcmI0ZXN0LmNvbTAPBgNVHRMBAf8EBTADAQH/MAsG
A1UdDwQEAwIBtjAdBgNVHQ4EFgQU+F+TquW0AGqCXM2kCHE6oWx8cPMwDQYJKoZI
hvcNAQEFBQADgYEAPsvB64aSX53UUXTSxuKBo8YuPKN9xb4iOtXRHSPJ/i2b2Jq
AVlv7ldpTfHzPYS6appt2Hue9GLxpWobSCarOLUUS5Bij4B7CbuNqrKAXMDhFoT0
wtcO9awwwr/cNKUjYRwpEYF+C6BpOMuPfp0zv1iLWYqZDs4DBxvN545l/Fk=
-----END CERTIFICATE-----
(ctrl-z)
#
# Host configuration
#
loadBalance host host_1 192.168.1.1 vRouter veritest:default
loadBalance host host_2 192.168.1.2 vRouter veritest:default
loadBalance host host_3 192.168.1.3 vRouter veritest:default

<repetitive entries for hosts 4 – 177 removed>

loadBalance host host_178 192.168.1.178 vRouter veritest:default
loadBalance host host_179 192.168.1.179 vRouter veritest:default

```

```

loadBalance host host_180 192.168.1.180 vRouter veritest:default
#
# Expressions used to classify the application data stream
#
loadBalance objectRule default {URI_PATH matches ""}
loadBalance objectRule jpg {URI_SUFFIX eq "jpg"}
#
# Real service parameters
#
loadBalance realService rs_clear_1 host_1 certType Literal
loadBalance realService rs_clear_1
#
# Real service advanced settings
#
advanced rcvWnd 65535 smmStreamLimit 4xRcvWnd
exit; exit;
loadBalance realService rs_clear_2 host_2 certType Literal
loadBalance realService rs_clear_2
#
# Real service advanced settings
#
advanced rcvWnd 65535 smmStreamLimit 4xRcvWnd
exit; exit;
loadBalance realService rs_clear_3 host_3 certType Literal
loadBalance realService rs_clear_3
#
# Real service advanced settings
#
advanced rcvWnd 65535 smmStreamLimit 4xRcvWnd
exit; exit;

<repetitive entries for realServices 4 – 177 removed>

loadBalance realService rs_clear_178 host_178 certType Literal
loadBalance realService rs_clear_178
#
# Real service advanced settings
#
advanced rcvWnd 65535 smmStreamLimit 4xRcvWnd
exit; exit;
loadBalance realService rs_clear_179 host_179 certType Literal
loadBalance realService rs_clear_179
#
# Real service advanced settings
#
advanced rcvWnd 65535 smmStreamLimit 4xRcvWnd
exit; exit;
loadBalance realService rs_clear_180 host_180 certType Literal
loadBalance realService rs_clear_180
#
# Real service advanced settings
#
advanced rcvWnd 65535 smmStreamLimit 4xRcvWnd
exit; exit;
#
# Request Policies
#
loadBalance requestPolicy QP_DEFAULT forward default sg_clear 10 optimizeLastResponse enabled firstObjectSwitching enabled
loadBalance requestPolicy QP_DEFAULT
exit; exit;
loadBalance requestPolicy QP_JPG forward jpg sg_clear optimizeLastResponse enabled firstObjectSwitching enabled
loadBalance requestPolicy QP_JPG
exit; exit;
#
# Service group configuration
#
loadBalance serviceGroup sg_clear roundRobin \
{rs_clear_1; rs_clear_10; rs_clear_100; rs_clear_101; rs_clear_102; rs_clear_103; rs_clear_104; rs_clear_105; rs_clear_106;
rs_clear_107; rs_clear_108; rs_clear_109; rs_clear_11; rs_clear_110; rs_clear_111; rs_clear_112; rs_clear_113; rs_clear_114;
rs_clear_115; rs_clear_116; rs_clear_117; rs_clear_118; rs_clear_119; rs_clear_12; rs_clear_120; rs_clear_121; rs_clear_122;
rs_clear_123; rs_clear_124; rs_clear_125; rs_clear_126; rs_clear_127; rs_clear_128; rs_clear_129; rs_clear_13; rs_clear_130;

```

```

rs_clear_131; rs_clear_132; rs_clear_133; rs_clear_134; rs_clear_135; rs_clear_136; rs_clear_137; rs_clear_138; rs_clear_139;
rs_clear_14; rs_clear_140; rs_clear_141; rs_clear_142; rs_clear_143; rs_clear_144; rs_clear_145; rs_clear_146; rs_clear_147;
rs_clear_148; rs_clear_149; rs_clear_15; rs_clear_150; rs_clear_151; rs_clear_152; rs_clear_153; rs_clear_154; rs_clear_155;
rs_clear_156; rs_clear_157; rs_clear_158; rs_clear_159; rs_clear_16; rs_clear_160; rs_clear_161; rs_clear_162; rs_clear_163;
rs_clear_164; rs_clear_165; rs_clear_166; rs_clear_167; rs_clear_168; rs_clear_169; rs_clear_17; rs_clear_170; rs_clear_171;
rs_clear_172; rs_clear_173; rs_clear_174; rs_clear_175; rs_clear_176; rs_clear_177; rs_clear_178; rs_clear_179; rs_clear_18;
rs_clear_180; rs_clear_19; rs_clear_2; rs_clear_20; rs_clear_21; rs_clear_22; rs_clear_23; rs_clear_24; rs_clear_25; rs_clear_26;
rs_clear_27; rs_clear_28; rs_clear_29; rs_clear_3; rs_clear_30; rs_clear_31; rs_clear_32; rs_clear_33; rs_clear_34; rs_clear_35;
rs_clear_36; rs_clear_37; rs_clear_38; rs_clear_39; rs_clear_4; rs_clear_40; rs_clear_41; rs_clear_42; rs_clear_43; rs_clear_44;
rs_clear_45; rs_clear_46; rs_clear_47; rs_clear_48; rs_clear_49; rs_clear_5; rs_clear_50; rs_clear_51; rs_clear_52; rs_clear_53;
rs_clear_54; rs_clear_55; rs_clear_56; rs_clear_57; rs_clear_58; rs_clear_59; rs_clear_6; rs_clear_60; rs_clear_61; rs_clear_62;
rs_clear_63; rs_clear_64; rs_clear_65; rs_clear_66; rs_clear_67; rs_clear_68; rs_clear_69; rs_clear_7; rs_clear_70; rs_clear_71;
rs_clear_72; rs_clear_73; rs_clear_74; rs_clear_75; rs_clear_76; rs_clear_77; rs_clear_78; rs_clear_79; rs_clear_8; rs_clear_80;
rs_clear_81; rs_clear_82; rs_clear_83; rs_clear_84; rs_clear_85; rs_clear_86; rs_clear_87; rs_clear_88; rs_clear_89; rs_clear_9;
rs_clear_90; rs_clear_91; rs_clear_92; rs_clear_93; rs_clear_94; rs_clear_95; rs_clear_96; rs_clear_97; rs_clear_98; rs_clear_99;
rs_clear_0) \
    inlineHealthCheck disabled
loadBalance serviceGroup sg_clear
exit; exit;
#
# Virtual Service Group Configuration
#
loadBalance vsGroup default {veritest-clear; veritest_secure}
#
# Virtual Service configuration
#
loadBalance virtualService veritest-clear HTTP 10.0.0.250 \
{QP_DEFAULT; QP_JPG}
loadBalance virtualService veritest-clear
#
# Virtual service advanced settings
#
advanced rcvWnd 65535 smmStreamLimit 4xRcvWnd initParseWithData true
exit; exit;
loadBalance virtualService veritest_secure HTTPS 10.0.0.250 {QP_DEFAULT; QP_JPG} ckmKeyName veritest sslCiphers \
{RSA_EXPORT_WITH_RC4_40_MD5; RSA_WITH_RC4_128_MD5; RSA_WITH_RC4_128_SHA;
RSA_EXPORT_WITH_DES40_CBC_SHA; RSA_WITH_DES_CBC_SHA; RSA_WITH_3DES_EDE_CBC_SHA;
RSA_WITH_AES_128_CBC_SHA; RSA_EXPORT1024_WITH_DES_CBC_SHA; RSA_EXPORT1024_WITH_RC4_56_SHA}
loadBalance virtualService veritest_secure
#
# Virtual service advanced settings
#
advanced rcvWnd 65535 smmStreamLimit 4xRcvWnd initParseWithData true
exit; exit;
#
# Port Bandwidth configuration
#
resource portBandwidth eth.1.1 100 100 65534 65535
resource portBandwidth eth.1.3 100 100 65534 65535
resource portBandwidth eth.1.4 100 100 65534 65535
resource portBandwidth eth.1.5 100 100 65534 65535
resource portBandwidth eth.1.8 100 100 65534 65535
resource portBandwidth eth.1.9 100 100 65534 65535
resource portBandwidth eth.1.12 100 100 65534 65535
#
# Service Engine Bandwidth Configuration
#
resource serviceBandwidth functionCard1 100 100
#
# vRouter configuration
#
vRouter default {Default vRouter}
vRouter default
#
# VLAN configuration parameters
#
vlan servers 20
vlan servers linkUpDownTrap disabled
vlan servers
#
# VLAN interface configuration

```

```

#
interface eth.1.4
interface eth.1.4 linkUpDownTrap disabled
interface eth.1.8
interface eth.1.8 linkUpDownTrap disabled
interface eth.1.12
interface eth.1.12 linkUpDownTrap disabled
#
# VLAN STP Interface Configuration
#
interface spanningTree eth.1.4 pathCost 4
interface spanningTree eth.1.8 pathCost 4
interface spanningTree eth.1.12 pathCost 4
exit;
#
# Display IP layer configuration
#
ip forwarding enabled
ip
#
# IP Interfaces
#
interface vlan.servers
interface vlan.servers linkUpDownTrap disabled

#
# IP Interface Address
#
address vlan.servers 192.168.1.254 255.255.255.0

#
# ICMP configuration
#
icmp true true false true
exit;
#
# Interfaces
#
interfaces sock.veritest:default linkUpDownTrap disabled
interfaces sock.veritest:default/ip.veritest:default linkUpDownTrap disabled
interfaces ip.veritest:default linkUpDownTrap disabled
interfaces ip.veritest:default/vlan.servers linkUpDownTrap disabled
interfaces vlan.servers linkUpDownTrap disabled
interfaces vlan.servers/eth.1.4 linkUpDownTrap disabled
interfaces vlan.servers/eth.1.8 linkUpDownTrap disabled
interfaces vlan.servers/eth.1.12 linkUpDownTrap disabled
interfaces loopback linkUpDownTrap disabled
#
# VRRP configuration
#
vrrp
exit;
exit;
#
# HTTP configuration and status
#
switchServices httpd enabled
switchServices httpd
exit; exit;
#
# Global NTP configuration parameters
#
switchServices ntp
#
# Global NTP configuration parameters (advanced)
#
advanced
exit;
exit; exit;

```

```

#
# SNMP configuration
#
switchServices snmp
#
# SNMP engine configuration
#
systemInfo {} {Sun Application Switch} {}
exit; exit;
#
# Software key
#
switchServices software key {}
#
# SSHd configuration and operation
#
switchServices sshd confEncryption {des3Cbc; blowfishCbc; des} confHmac \
{md5; sha1; md5b96; sha1b96} userAuthentication {publicKey; password}
switchServices sshd
#
# SSHd configuration and operation (advanced)
#
advanced
exit;
exit; exit;
#
# Telnetd configuration and current status
#
switchServices telnetd enabled
switchServices telnetd
exit; exit;
#
# TFTPd configuration and session statistics
#
switchServices tftpd
exit; exit;
#
# TideRunner Configuration
#
switchServices tideRunner initkeys functionCard1 20000 statPollPeriod 5 smmPageSize 8 dleMaxHdrLen 8192
#
# Configuration and status for the trap process
#
switchServices trap
exit; exit;

```

VeriTest ([www.veritest.com](http://www.veritest.com)), the testing division of Lionbridge Technologies, Inc., provides outsourced testing solutions that maximize revenue and reduce costs for our clients. For companies who use high-tech products as well as those who produce them, smoothly functioning technology is essential to business success. VeriTest helps our clients identify and correct technology problems in their products and in their line of business applications by providing the widest range of testing services available.

VeriTest created the suite of industry-standard benchmark software that includes WebBench, NetBench, Winstone, and WinBench. We've distributed over 20 million copies of these tools, which are in use at every one of the 2001 Fortune 100 companies. Our Internet BenchMark service provides the definitive ratings for Internet Service Providers in the US, Canada, and the UK.

Under our former names of ZD Labs and eTesting Labs, and as part of VeriTest since July of 2002, we have delivered rigorous, objective, independent testing and analysis for over a decade. With the most knowledgeable staff in the business, testing facilities around the world, and almost 1,600 dedicated network PCs, VeriTest offers our clients the expertise and equipment necessary to meet all their testing needs.

**For more information** email us at [info@veritest.com](mailto:info@veritest.com) or call us at 919-380-2800.

#### **Disclaimer of Warranties; Limitation of Liability:**

VERITEST HAS MADE REASONABLE EFFORTS TO ENSURE THE ACCURACY AND VALIDITY OF ITS TESTING, HOWEVER, VERITEST SPECIFICALLY DISCLAIMS ANY WARRANTY, EXPRESSED OR IMPLIED, RELATING TO THE TEST RESULTS AND ANALYSIS, THEIR ACCURACY, COMPLETENESS OR QUALITY, INCLUDING ANY IMPLIED WARRANTY OF FITNESS FOR ANY PARTICULAR PURPOSE. ALL PERSONS OR ENTITIES RELYING ON THE RESULTS OF ANY TESTING DO SO AT THEIR OWN RISK, AND AGREE THAT VERITEST, ITS EMPLOYEES AND ITS SUBCONTRACTORS SHALL HAVE NO LIABILITY WHATSOEVER FROM ANY CLAIM OF LOSS OR DAMAGE ON ACCOUNT OF ANY ALLEGED ERROR OR DEFECT IN ANY TESTING PROCEDURE OR RESULT.

IN NO EVENT SHALL VERITEST BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH ITS TESTING, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL VERITEST'S LIABILITY, INCLUDING FOR DIRECT DAMAGES, EXCEED THE AMOUNTS PAID IN CONNECTION WITH VERITEST'S TESTING. CUSTOMER'S SOLE AND EXCLUSIVE REMEDIES ARE AS SET FORTH HEREIN.