

WEBSTAR

**Web Application
Stress Test
and Data Analysis**

April 15, 2002
by Khalid Anwar and Arif Saleem

Table of Contents

1	WEB APPLICATION STRESS TESTING	1-3
1.1	Test environment	1-3
1.1.1	Hardware Platform Configurations	1-3
1.2	Web Stress Test Plan	1-4
1.2.1	Web Application Stress Tool	1-4
1.3	Typical Trial Report	1-8
2.	Data Analysis	1-10
2.1	Maximum Requests per Second	1-10
2.2	Maximum Average Load on the server	1-12
2.3	Response Time	1-13
2.3.1	Time To Last Byte (TTLB)	1-13
2.3.2	Time To First Byte (TTLB)	1-13
2.4	Socket Connects	1-13
2.5	Failure	1-14
2.6	Bytes Sent Rate/ Bytes Receive Rate	1-14
3.	SUMMARY OF REPORT	1-15

LIST OF FIGURES

Figure 2-1.	Application Settings	1-5
Figure 2-2.	Typical Trial Summary Report	1-9
Figure 2-3.	Number of Threads and RPS (POST method, No BW throttling)	1-10
Figure 2-4.	Number of Threads and RPS (GET method, No BW throttling)	1-11
Figure 2-5.	Number of Threads and RPS (56K BW throttling)	1-11

LIST OF TABLES

Table 2-1.	Client Test Configurations	1-7
------------	----------------------------	-----

Abstract

The objective of this test was to assess and analyze the scalability and performance of a web application that used PHP scripts running on Sun Solaris 7 using Apache web server and Oracle database. The specific goals included the following:

- Address possible causes of performance bottlenecks.
- Determine how fast users can expect web pages to be returned to them.

We used Microsoft's Web Application Stress Tool, WAS, to meet these objectives.

This paper presents the process and results of our findings. We tested the web application on a Netra T1 1440, Solaris 7 running Apache 1.3.12. A Sun Ultra 10 running Solaris 7 and Oracle 8.1.7 was used as the database server. We collected stress data and other statistics, including requests per second (RPS) and time to last byte (TTLB). The maximum RPS data infers the number of concurrent users that a server platform can support. The TTLB data shows how fast the results are presented to the client by the server.

The test environment is described in Section 1.1, and the test results are presented in Section 2.2. Section 2.3 explains the data analyses.

1 Web Application Stress Testing

1.1 Test environment

1.1.1 Hardware Platform Configurations

The test was performed in a lab environment, on an isolated 100BaseT local area network (LAN), giving a maximum bandwidth of 100Mbps/s. The PHP scripts were run on the Netra T1 1440 Server, with Solaris 7 operating System and Apache 1.3 web server. The Netra had a single 440MHz UltraSparc CPU and 1GB RAM. The Oracle 8.1.7 database was run on a Sun Ultra 10 server, also running Solaris 7, with a single 440MHz UltraSparc CPU and 512MB RAM.

1.1.2 User Accounts and Client Machine

A maximum of 250 user accounts were established on the WAS client machines to simulate access to the servers.

The WAS tool client was installed on a Microsoft Windows 2000 workstation with a 533-Mhz processor and 187 MB of RAM. To stress a powerful server, either one powerful client or several less powerful clients are set up to provide the load.

We determined the maximum number of requests per second that the Web server can handle, the number of concurrent users when failures start, bytes sent and received rates, the maximum average load on the server; we discussed server resource bottlenecks and presented system response times. Section 2.1 describes how the tests were performed, Section 2.2 presents a typical test result, and Section 2.3 presents the conclusions of our Web application stress tests.

1.2 Web Stress Test Plan

1.2.1 Web Application Stress Tool

The Microsoft Web Application Stress (WAS) tool is designed to simulate multiple browsers requesting pages from a Web site. This tool can realistically simulate many requests with relatively few client machines. Microsoft's tutorial on the WAS tool includes information on installation, scripting, performance counters, settings, and reporting. It also provides general guidelines for using WAS and describes common Web testing problems. The tutorial is located at the WAS tool site (<http://webtool.rte.microsoft.com>). The following paragraphs describe our setup to perform these tests.

1.2.1.1 Application Settings

Figure 2-1 shows the settings we used to perform the tests. All variables in the settings are kept constant except the Concurrent Connections (or number of users) setting, which includes Stress Level and Stress Multiplier. The Stress Level (threads) times the Stress Multiplier (sockets per thread) represents the stress level for each system. These settings were changed for every trial of the tests. All tests were run for 2 minutes, in order to allow the system load to stabilise. We observed the number of hits on each page for each test run. The number of hits on each page is shown in Figure 2-2, Typical Trial Summary Report. Request delays were not used in the tests, as we aimed to stress the server.

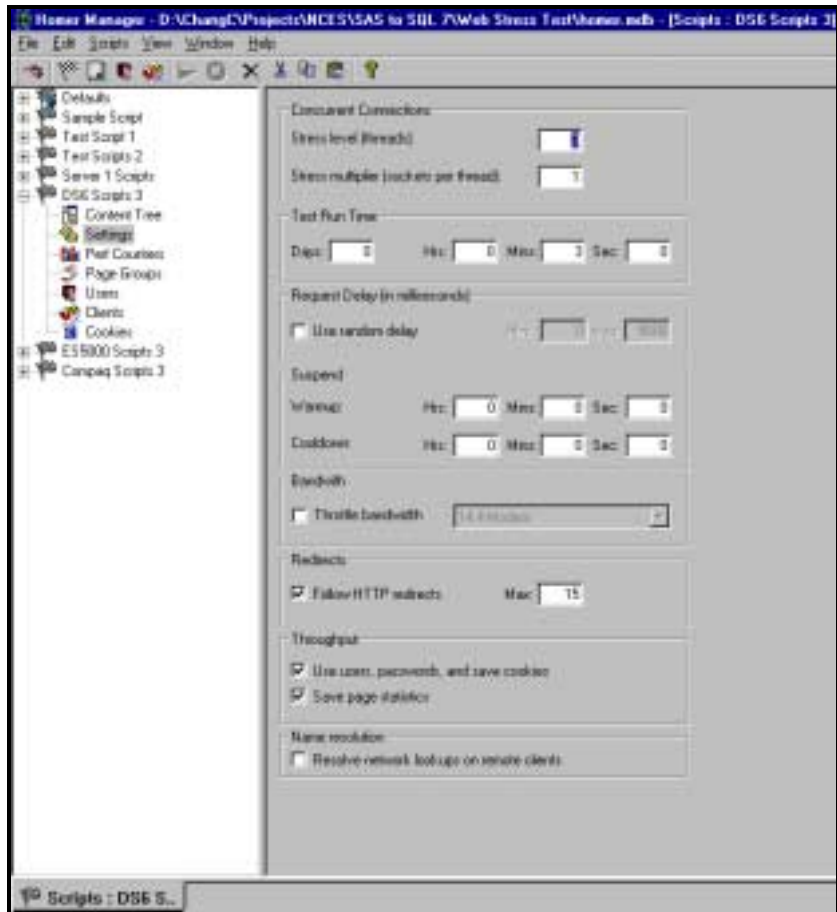


Figure 2-1. Application Settings

1.2.1.2 Test Scripts

Test were performed using three sets of PHP scripts running on the Netra Web server (URL base – a section of <http://www.pharmalife.co.uk>):

Signup scripts (Script Type A)

- database-driven PHP scripts with an HTML form.

News Home Page scripts (Script Type B)

- database-driven PHP scripts

Terms and Conditions script (Script Type C)

- HTML script with embedded PHP code but no database accesses.

Test Scenario

Test was performed on two scenarios:

- With Bandwidth (BW) throttling (56 K Modem)

 'Sign-up' scripts (Form method POST) –13 trials

 'News Home Page' script – 13 trials

- Without Bandwidth (non-BW) throttling

 'Terms & Conditions' script (HTML) – 13 trials

 'Sign-up' scripts

 Form method: POST – 13 trials

 Form method: GET – 13 trials

For each test, the average load on the server was noted at intervals of 6 seconds. The total test duration was kept constant at 2 minutes.

1.2.1.3 Reporting - overview

Definitions of terms used :

Number of hits: The total number of requests made by Microsoft Web Application Stress during this test run.

Requests per second: The average requests per second throughout the test.

Socket errors: *Connect* – The number of times any client failed to connect to the web server.

Send – The number of times any client failed to send data to the web server.

Recv – The number of times any client failed to receive data from the web server.

Timeouts: The number of threads that timed-out, and subsequently were closed.

TTFB Avg: *Time To First Byte Average.* The total time, in milliseconds, from the initial request until the first byte of data is received on the client. This is an average over the test.

TTLB Avg: *Time To Last Byte Average.* The total time, in milliseconds, from the initial request until the last byte of data is received on the client. This is an average over the test.

The TTFB percentiles calculate the time from the request for the page until Web Application Stress receives the first byte of data, in milliseconds. All of the requests are sorted, and then the data is divided into percentiles.

The TTLB percentiles calculate the total time from the request until the last byte of data has been received on the client, in milliseconds. This number includes the TTFB time and any additional time needed to receive the last byte of data. All of the requests are sorted, and then the data is divided into percentiles.

Table 2-1. Client Test Configurations

Test sequence	Script	Bandwidth Throttling	Maximum connections	Run Time duration (Minutes)
1	Signup (Using POST method)	56 K Modem	250	2
2	News Home Page	56 K Modem	250	2
3	Terms and Conditions	No BW throttling	250	2
4	Signup (Using GET method)	No BW throttling	250	2
5	Signup (Using POST method)	No BW throttling	250	2

1.3 Typical Trial Report

A WAS trial report contains nine sections: Summary, Overview, Script Settings, Test Clients, Result Codes, Page Summary, Page Groups, Page Data, and Performance Counters. The Summary report presents a great deal of vital information regarding the trial. A Typical Trial Summary Report is shown in Figure 2-2. To examine trial details, one can review the results from the other eight sections of the report.

To validate a trial, the report on the Result Codes is reviewed first. This example indicates the clients received all pages.

Result Codes		
Code	Description	Count
=====		
200	OK	846

However, the example below indicates 952 pages were received, but 39,042 pages were not found. These results indicate that this is not a valid trial, something is not set up correctly, and troubleshooting will be necessary.

Result Codes		
Code	Description	Count
=====		
200	OK	1402
404	Not Found	1880

This report includes only those trial results that are valid.

“Requests per Second” and “Time to Last Byte (TTLB)” (bolded in Figure 2-2 below) are addressed in detail in paragraph X.X, which discusses the data analysis. Performance monitor results are not presented in the summary reports. A sample Performance Counters output is shown in the next section.

Overview	
=====	
Report name:	15/04/2002 13:08:16
Run on:	15/04/2002 13:08:16
Run length:	00:02:00
Web Application Stress Tool Version:	1.1.293.1
Number of test clients:	1
Number of hits:	830
Requests per Second:	6.91

Socket Statistics

Socket Connects: 837
Total Bytes Sent (in KB): 745.45
Bytes Sent Rate (in KB/s): 6.20
Total Bytes Recv (in KB): 27050.20
Bytes Recv Rate (in KB/s): 225.10

Socket Errors

Connect: 0
Send: 0
Recv: 0
Timeouts: 0

RDS Results

Successful Queries: 0

Script Settings

=====
Server: www.pharmalife.co.uk
Number of threads: 20

Test length: 00:02:00
Warmup: 00:00:00
Cooldown: 00:00:00

Use Random Delay: No

Follow Redirects: Yes
Max Redirect Depth: 15

Clients used in test

=====
localhost

Clients not used in test

Result Codes

=====
Code Description Count

200 OK 830

Page Summary

=====
Page Hits TTFB Avg TTLB Avg Auth Query

POST /signup script 830 1842.44 2847.28 No Yes

Figure 2-2. Typical Trial Summary Report

2. Data Analysis

The first goal of this analysis was to find the maximum requests per second that the Web server can handle. The second goal was to determine the possible cause of any performance bottlenecks, such as CPU, memory, or backend dependencies. The TTLB data, which show how fast a server can present the data to clients, are also presented in this section.

2.1 Maximum Requests per Second

What are the “maximum requests” a server can handle? The “requests per second” are noted for each trial load. An increased server stress load is accomplished by increasing the Stress Level (threads) setting. The data for number of threads and requests per second are presented in the three figures below for three different scripts.

Figure 2-4 shows 13 trials of the POST method. The load, or number of threads, increased from 1 to 250, and the requests per second (RPS) leveled off at about 6.6.

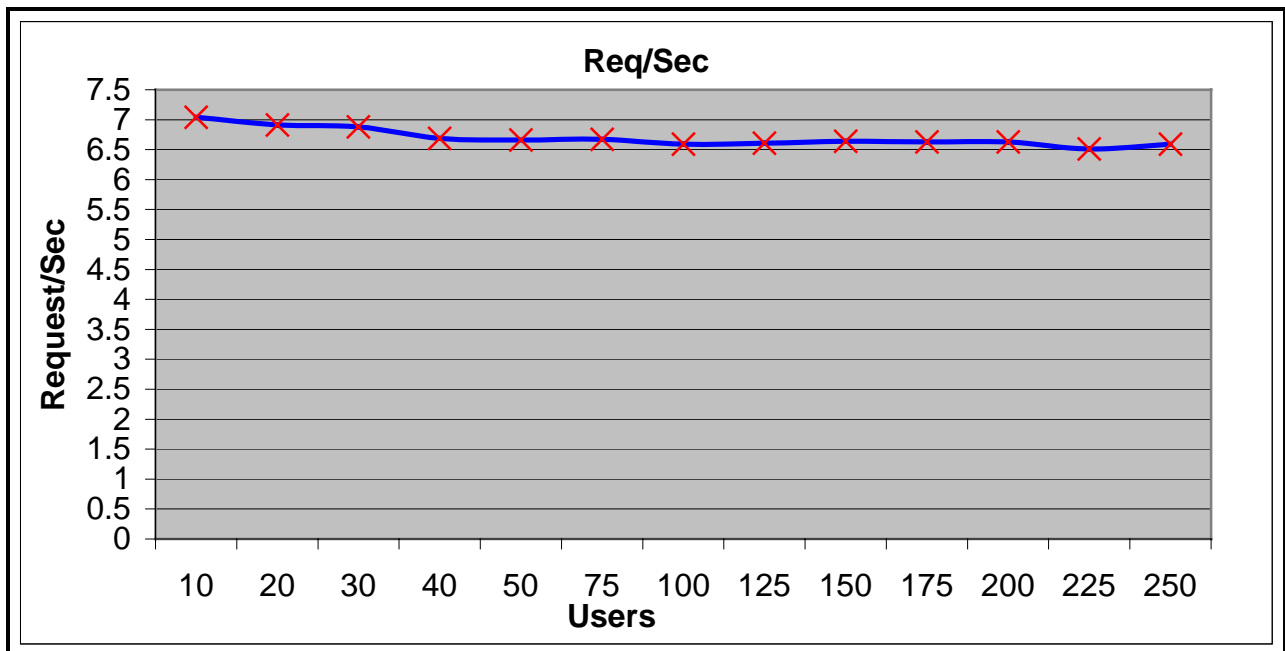


Figure 2-3. Number of Threads and RPS (POST method, No BW throttling)

Figure 2-4 shows 13 trials of the of the GET method. The load, or number of threads, increased from 1 to 250, and the RPS leveled off at about 6.7. Note the decrease in the Request per second value after the concurrent connection increases beyond 20.

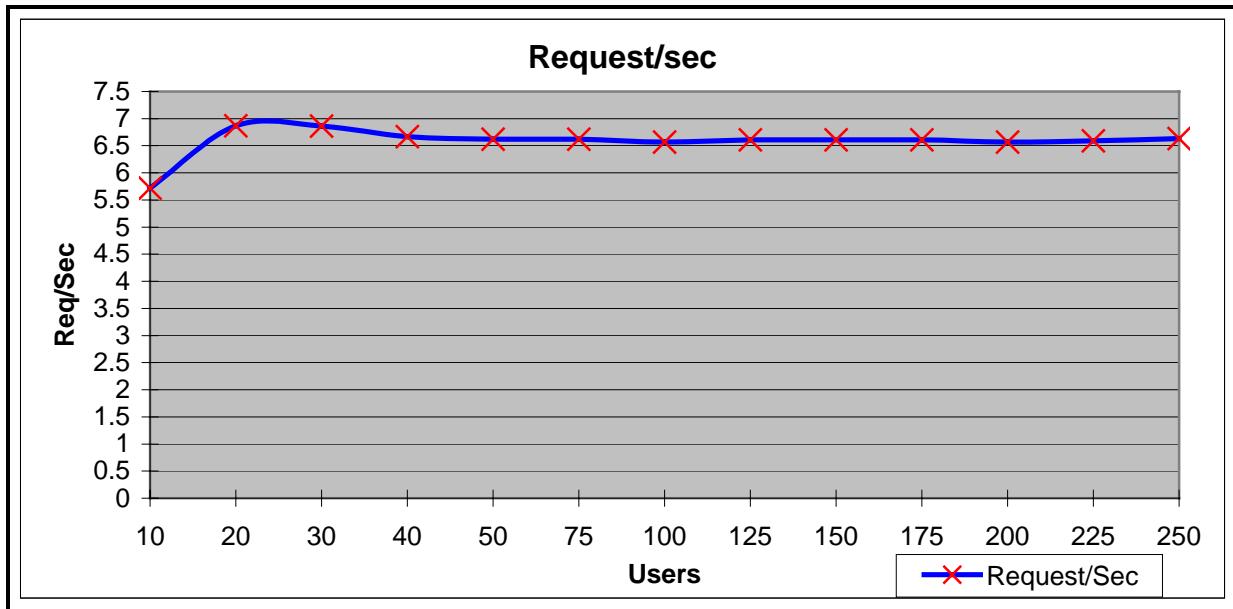


Figure 2-4. Number of Threads and RPS (GET method, No BW throttling)

Figure 2-5 shows 13 trials of a simple script run for 2 minutes with Bandwidth throttling of 56 K. The load, or number of threads, increased from 1 to 250, and the RPS decreased linearly to just above 7.

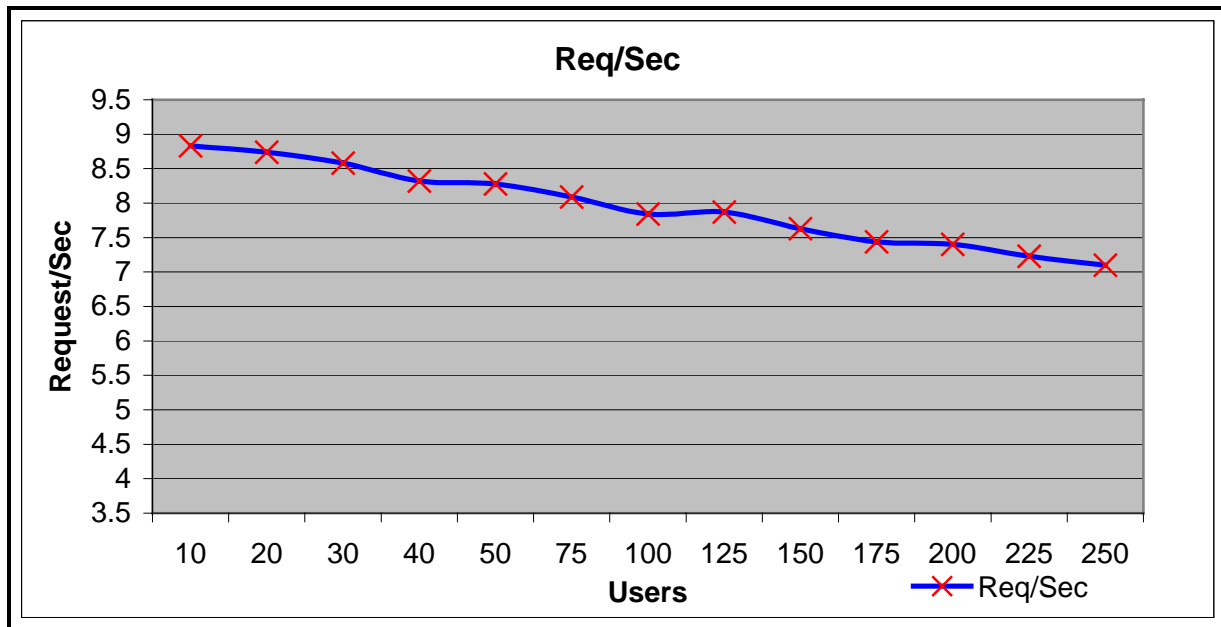


Figure 2-5. Number of Threads and RPS (56K BW throttling)

In conclusion, the figures 2.3 and figure 2.4 clearly show that the Server can handle a maximum of 6.5 to 6.6 requests per second, when there is no BW throttling. It is also seen that the server gives the same RPS value for both the POST and GET scripts.

On the other hand, when we restrict the BW to 56K, we obtain a higher RPS value, as compared to the non-BW throttled test. The RPS value decreases as the stress level is increased (concurrent connection is increased). The reason for this is that initially the individual BW of the client decides the RPS value but as the load on the server is increased, the server uses up all the allotted BW and hence is unable to respond to more requests.

How do these maximum RPS values translate into the number of users we can support per server? The RPS is directly correlated to the number of users, but the coefficient of this correlation is unknown. This coefficient will vary with different web environments, bandwidth environments, and user access behaviors. A Web server may contain many static pages that are less costly to system resources than PHP scripts. A Web server may receive requests from high bandwidth broadband connections and from modem dialups – hence different clients have different bandwidth limitations. Users do not typically generate continuous requests; ordinarily, they request a page, read it, and then request another page. The request-and-reading-time ratio will depend on the Web server page type and users' access behavior.

2.2 Maximum Average Load on the server

From the charts plotted from the test data, we found that the maximum average load value for the database-driven PHP scripts is higher than for simple HTML pages. The results show that this value increases for scripts which submit form data. The system load value settles to approximately 43 for the simple script whereas it varies from 46 to 48 for other tests. The load value becomes almost steady after 75 connections. The other charts show that the values of bytes received rate, bytes sent rate, number of request per second, and number of hits, starts decreasing or becomes almost steady after this value. These values also depend upon a number of other factors such as bandwidth throttling, time delays etc. The maximum average load value indicates that after 75 connections, the system begins to fail processing some requests, even though the TCP connection is made between the client and the server. This is more clear from the Socket connects graph. A study of this graph determines the behavior of the server with respect to the number of connections.

2.3 Response Time

2.3.1 Time To Last Byte (TTLB)

Connections	Signup (GET method)	Signup (POST Method)	Terms and Condition(56K)	News Home (56k)
10	1452.64	1114.33	1404.24	3199.21
20	2859.96	2234.08	2847.28	3618.51
30	4277.79	3431.77	4242.44	5533.31
40	5810.14	4696.03	5804.44	7615.17
50	7305.81	5907.39	7287.24	9587.88
75	10854.23	8904.56	10817.69	14373.39
100	14219.59	12027.06	14249.3	19209.22
125	17570.6	15113.87	17510.98	23813.39
150	20555.91	18157.55	20556.36	28556.1
175	22628.04	20534.14	22676.88	32041.01
200	23235.33	21639.2	23133.07	34526.5
225	23502.4	22332.18	23413.14	35725.88
250	23530.51	23868.47	23505.44	37544.44

2.3.2 Time To First Byte (TTLB)

Connections	Signup (GET method)	Signup (POST method)	Terms and Condition(56K)	News Home (56K)
10	809.31	767.48	724.86	1069.75
20	1852.79	1842.44	1592.88	1911.63
30	3010.52	3008.18	2620.26	3008.14
40	4260.32	4231.3	3576.01	4233.64
50	5302.85	5305.68	4567.87	5478.06
75	8425.6	8400.7	7127.44	9394.39
100	11750.13	11708.26	9766.77	13033.02
125	14996.84	15015.04	12178.24	16394.69
150	18107.92	18059.5	14448.11	19317.87
175	20243.86	20214.71	15967.14	21153.75
200	20721.55	20790.46	16153.05	20894.38
225	21021.06	20948.86	15988.13	20439.91
250	21078.49	21046.5	15931.21	19846.16

2.4 Socket Connects

A comparison of this graph with BW limited versus no BW limit, depicts that the socket connection value increases for a BW limited test. From this we conclude that the socket connections depend on the BW throttling.

2.5 Failure

The failure starts after 175 concurrent connections for all scenarios.

2.6 Bytes Sent Rate/ Bytes Receive Rate

This value depends on the BW throttling and the point when failure starts.

	GET	POST	BW Throttling
Maximum average load	46.5-48.8	46-48.7	42.5-43.5
Hits	790-802	792-802	1053-846
Socket Connects	845-795-853	852-791-834	1064-1123
Failure	175	175	175
Request/Second	6.60-6.67	6.60-6.69	8.83-7.1
Bytes sent rate(kb/s)	Decreases up to 75 connections then increases.	Decreases up to 75 connections then increases.	Decreases
Bytes receive rate(kb/s)	Decreases up to 40 connections then becomes steady	Decreases up to 40 connections then becomes steady	Decreases
TTFB	770	767	724
TTLB	1410	1404	1114

3. Summary of Report

- Average maximum load settles off after 75 connections in all scenarios.
- The number of hits reduces after 30 users in all scenarios.
- Request per second falls beyond 30 users. This value is greater for a static page (no database processing).
- The value of bytes sent rate remains almost constant until the failure starts.
- Bytes receive rate always decreases as the number of users increases.
- TTFB and TTLB values vary linearly until failure starts.
- There is no significant difference between a 'no bandwidth limitation' and 'bandwidth limitation' scenarios. The slight difference in the two scenarios could be due to the server 'dropping' some requests at very high number of concurrent users, freeing up resources to deal with the requests that are accepted.
- The system gets severely overloaded after 20 concurrent connections.
- Response time is dependent upon the number of concurrent connection.
- Using GET or POST as the HTTP submission method does not affect the load on the server or the request per second.
- In all scenarios, failures began at about 175 concurrent users, indicating that even a low power Netra with a single CPU can serve a fairly large website.

Conclusions

The conclusions derived relate to two main issues: firstly, the issue of bandwidth, and secondly the issue of system load.

In the bandwidth limited tests, with a small number of concurrent users (upto 20) the bottleneck was found to be not so much the server, but rather the clients.

Hence, in a scenario where most of the users are using modems, the server will be able to serve a larger number requests.

However, as the number of concurrent users increases, the server begins to get loaded similarly to the non-bandwidth limited case.

At very high numbers of concurrent users, the non-bandwidth limited case actually improves, while the BW-limited case remains constant. This could be because the server begins to 'drop' some requests, freeing up resources to deal with the requests that are accepted.

In all scenarios, we found that the maximum load levels off at about 43. The only exception was in the case of the PHP scripts that submitted form data to the database – suggesting that such scripts require more processing on the server.

From the form submission tests, we also find that using GET or POST as the HTTP submission method does not affect the load on the server or the RPS.

With respect to system load, in order to understand some of the implications of these results we need to explain two terms in particular - '**load**' and '**concurrent users**'.

A *load* value of 1.0 on a server roughly indicates that the resources of the server are fully utilised. This load value is calculated from many parameters - such as CPU utilisation, memory utilisation, swapping to disk ('disk thrashing'), network connections, etc. Hence it is possible that one factor - say network connections - can cause an apparently very high load value, while other parameters such as CPU usage are very low.

However, in general, a load of 2.0 would suggest that a machine with twice as many resources would be needed to satisfactorily meet the demand laid upon the system.

'*Concurrent users*' in the tests refers to the case where two or more simultaneous requests for web pages - which could be different - are sent to the server. This corresponds to the case where multiple users enter a URL in their browsers *at the same time*.

This is different from '*actively browsing users*', as in the second case multiple users may be browsing through the site at the same time but actually clicking on links, to

request pages, at *different* times. This is because users request a page, then spend some time reading it before clicking a link to request another page.

The 'actively browsing users' is again different to the '*user base*' of the Web Site, which refers to the Registered users of the web site - only some of whom may be online at a given time.

Thus, the 'user base' may be, say, a thousand, of whom at any given time 100 may be 'actively browsing' the site. Of these 100, only 5 may actually concurrently request pages from the server at any given time.

In our tests on the Netra, a 'concurrent user value' of 1 would imply a number of 'actively browsing users' such that a new request arrives at the servers as soon as the last one has been completed. A 'concurrent user value' of 5 would indicate a much larger number of 'active users' perhaps a few hundred. Hence, the tests which are performed at concurrent user levels above 20 do not really represent a 'real scenario' but are merely useful to see how far the machine can be stressed.

Thus, in our tests, we find that in those cases where there is only 1 concurrent user, the system load remains around 1.0. This shows that the Netra is only powerful enough to comfortably handle a single continuous stream of requests. More than this - that is 2 or more concurrent users - takes the load level above 1.0, indicating that a load balanced web server solution would be required.

The maximum load reached rises roughly linearly as the number of concurrent users increases, until it levels off at a value of about 45 for more than 50 concurrent users. Also, we find that failures begin after about 175 concurrent users.

However, throughout our testing the Netra never crashed, showing the stability of the platform while under considerable stress.