

May 1999
ECG506/0599

Prepared by Internet and
E-Commerce Solutions Business
Unit

Enterprise Solutions Division

Compaq Computer Corporation

Contents

Introduction.....3
Problem Analysis.....3
Tested Configuration.....4
Overview of the Testbed5
Performance Testing6
 Test Methodologies.....7
Performance Optimization7
 Hardware and Operating
 System7
 Runtime.....8
Test Results10
 Static Performance: Network
 Bandwidth and RAM10
 Static Performance:
 Workload Size16
 Static Performance:
 Recommendations16
 CGI Performance: Overhead ...16
 CGI Performance: Additional
 Processor17
 CGI Performance:
 Recommendations18
**Appendix: Source Code for
Moderate CGI Program.....19**

Performance Characterization and Tuning of Apache Web Server

Abstract: This guide provides developers and system integrators with details of performance characterization and tuning for Linux in conjunction with the Apache HTTP Server on Compaq server hardware.

Notice

The information in this publication is subject to change without notice and is provided "AS IS" WITHOUT WARRANTY OF ANY KIND. THE ENTIRE RISK ARISING OUT OF THE USE OF THIS INFORMATION REMAINS WITH RECIPIENT. IN NO EVENT SHALL COMPAQ BE LIABLE FOR ANY DIRECT, CONSEQUENTIAL, INCIDENTAL, SPECIAL, PUNITIVE OR OTHER DAMAGES WHATSOEVER (INCLUDING WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION OR LOSS OF BUSINESS INFORMATION), EVEN IF COMPAQ HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The limited warranties for Compaq products are exclusively set forth in the documentation accompanying such products. Nothing herein should be construed as constituting a further or additional warranty.

This publication does not constitute an endorsement of the product or products that were tested. The configuration or configurations tested or described may or may not be the only available solution. This test is not a determination of product quality or correctness, nor does it ensure compliance with any federal, state or local requirements.

Compaq, ActiveAnswers, Deskpro, Compaq Insight Manager, Faststart, Systempro, Systempro/LT, ProLiant, ROMPaq, QVision, SmartStart, NetFlex, QuickFind, PaqFax, and Prosignia are registered with the United States Patent and Trademark Office.

Netelligent, Systempro/XL, SoftPaq, QuickBlank, QuickLock are trademarks and/or service marks of Compaq Computer Corporation.

Linux is a registered trademark of Linus Torvalds.

Pentium is a registered trademark of Intel Corporation.

Other product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

Performance Characterization and Tuning of Apache Web Server
Technical Guide prepared by Internet and E-Commerce Solutions Business Unit

Enterprise Solutions Division

First Edition (May 1999)

Document Number ECG506/0599

Introduction

This document provides guidelines for optimizing Apache Web Server on Linux for best performance. Although Compaq used the SuSE 6.0 Linux distribution to test these performance optimizations, these guidelines apply to the Linux kernel and to the Apache HTTP Server and can therefore be applied to any Linux distribution.

Problem Analysis

When analyzing a typical Web server's capability to handle work, Compaq determined that the major issues are the data throughput and the number of requests the server can handle in a given time period. Based on the results of testing the Apache HTTP Server on Linux, Compaq has identified the amount of RAM available to the server and CPU performance as the two most important factors determining this server's capacity to handle work.

Compaq set up a test environment that simulated thousands of requests coming into the Apache HTTP Server system and increased the demands on the server until the CPU utilization was maximized. This provided empirical data that allowed Compaq to identify the server's peak performance.

Compaq used these data to create the Compaq Sizer for Apache Web Server on Linux. The sizer uses the performance data with information supplied by the customer to recommend a hardware configuration that meets the customer's Web server requirements.

The sizer is available from the *Compaq ActiveAnswers* website at:
<http://www.compaq.com/ActiveAnswers/>


Tested Configuration

Although there are a number of ways to examine a Web server's performance and capability to handle work, the Compaq approach was limited to analyzing the total amount of work that can be accomplished by a server in a given amount of time. This approach identifies the peak capabilities of a server given a fixed system resources.

The empirical basis for conclusions drawn from this testing and for the data used to create the sizer is derived from performance testing conducted on the *Compaq ProLiant 1850R* server in a standalone configuration. Table 1 describes this server:

Table 1: Tested Configuration

ProLiant 1850R



Pentium II, 450MHz, 512 KB cache, 1P and 2P
128MB, 256MB, and 512 MB of DRAM
Integrated 10/100 TX UTP controller
Compaq PCI Netelligent Intel NIC
1x9.1 GB, 10000 RPM, SCSI hard drive (for the OS)
1x4.3 GB, 10000 RPM, SCSI hard drive

Overview of the Testbed

The server described in table 1 was placed as a system unit under test (SUT) into the test bed shown in figure 1. Compaq ran a defined suite of tests on the SUT and recorded the results for analysis. The server was configured with two Network Interface Controllers, each of which was connected to a 100baseTx full-duplex network.

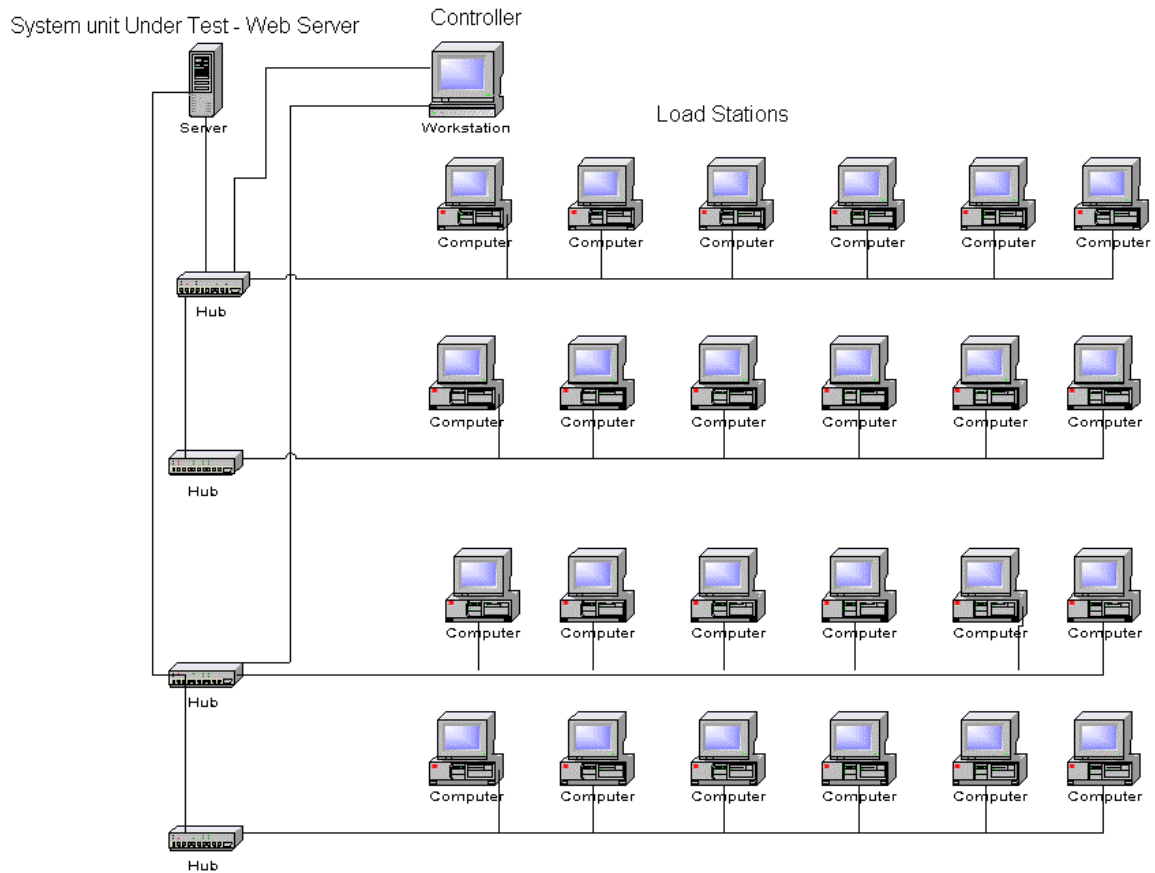


Figure 1. Testbed configuration.

Performance Testing

Compaq carried out the performance testing using a *Compaq ProLiant* 1850R server running SuSE Linux 6.0 and Apache HTTP Server 1.3.4. The Linux distribution was updated to use the 2.2.4 kernel, the latest kernel available.

Compaq made a number of modifications (described below in the section on "Performance Optimization") to the 2.2.4 kernel. These modifications are available¹ as a `.config` file that the user can place in the `/usr/src/linux` directory for the 2.2.4 kernel.

Compaq disabled most system services: only `crond`, `klogd`, `httpd`, `inetd`² and other necessary daemons (such as the console login daemons) were running. In this way, the testing focused on the Linux and Apache products rather than the peculiarities of a particular distribution.

Compaq selected the test methodologies based on an analysis of existing websites and their file content mix. File store sizes³ were 64MB, 256MB, and 512MB; file content varied between 100% percent static files, 100% simple CGI⁴ files, and 100% moderate CGI files. The file content included:

- A static-only file mix that consisted of the HTML file tree from the Web Bench 2.0 suite.
- A simple program mix that consisted of a CGI program echoing CGI variables back to the Web client. This consisted of several copies of the `simcgi` program deployed in a dense file system tree (with many sub-directories, each containing a few copies of the file) from the Web Bench 2.0 suite.
- A moderate CGI program that echoed CGI variables back to the Web client and wrote them to the file system in a file whose name depended on the process ID of the CGI program. To test different workload sizes, several copies of this file were made in a dense file system tree until the appropriate work load size was reached. The source code for this program is included in the appendix to this guide.

Note: To achieve larger file store sizes, Compaq simply replicated the original file tree so that multiple copies resided on the server.

¹ Use the appropriate link on the *ActiveAnswers* page from which this document was downloaded.

² Compaq left `inetd` running even though many users may disable this daemon for security reasons.

³ The average file size was 6kB.

⁴ Dynamic content.

Test Methodologies

Compaq tested each hardware configuration using an HTTP load test tool (Ziff-Davis WebBench 2.0) to apply an appropriate⁵ load for the particular test run.

Compaq ran each configuration/CPU-utilization combination three times. There were two purposes for this:

- To verify that one particular set of test results was comparable to another to eliminate any aberrations in the test data.
- To increase the number of data points available for analysis.

The basic test procedure included:

- Prior to testing a new configuration, all clients and servers were rebooted⁶.
- Each test was run for 10 minutes (a 4-minute ramp-up, a 5-minute run, and a 1-minute ramp-down).
- A `vmstat` job was running on the Web server to gather performance data.
- After each test run, a copy of the Ziff-Davis WebBench 2.0 performance data log was saved to hard disk for subsequent analysis.

Performance Optimization

Compaq determined that additional performance improvements and application scaling can be accomplished through hardware and operating system optimizations and runtime optimizations.

Details of the specific optimizations are given below.

Hardware and Operating System

To optimize Web server performance, Compaq provides these recommendations:

- **RAM:** Compaq determined that when the content being served is principally static, the single most important hardware factor affecting Web server performance is the amount of RAM.
- **Maximum number of clients:** Control the `MaxClients` setting of the Apache HTTP Server so that the web server can accommodate enough concurrent connections to maximize web server performance. Specific disclosure of values is made in the next section.
- **TCP/IP patches:** Use a recent kernel version to ensure that the latest Linux TCP/IP patches are installed. Version 2.2.3 (and later) contains several networking improvements over previous versions. Compaq used kernel version 2.2.4 was used for this testing.

⁵ Sufficient to drive the server to 100 percent CPU utilization.

⁶ Reboots occurred only when changing test configurations. When repeating a configuration the second and third time, there was no reboot.

Runtime

The runtime optimizations include optimizations recommended by Compaq, optimizations recommended by Apache, and updates to default settings.

Compaq Recommendations

To optimize Web server performance, Compaq provides these recommendations:

- **Removing non-essential daemons:** Use the appropriate Linux distribution tool to remove all unnecessary services so that only `crond`, `klogd`, `httpd`, `inetd` and other necessary daemons (such as the console login daemons) are running. For SuSE, the YaST tool is useful; Caldera provides a tool known as LISA in their OpenLinux version 1.3; Red Hat depends on the `linuxconfig` toolkit. Alternatively, the user can manually modify the `/etc/rc.d`, `etc/rc.d/init.d`, and `/etc/sysconfig` directories.
- **Increase the maximum number of files and inodes that the system can open at one time:** Add the following to the system's start-up scripts in order to allow the Web server to utilize as many of the system's resources as possible:

```
echo 16384 > /proc/sys/fs/file-max
echo 49152 > /proc/sys/fs/inode-max
```

The `/proc` file system in Linux provides an interface between the user and the kernel. Several read-only files allow the user to view kernel utilization statistics and are used by programs like `top` and `vmstat` to obtain the statistics that they present to the user.

Other writable files allow the user to change kernel parameters on the fly. These files (including those added to the start-up scripts above) allow the user to specify the maximum number of files and inodes that the kernel can have open at any given time. The suggested values are about 4 times the default values for these parameters for kernel version 2.2.4.

IMPORTANT: This single optimization resulted in a nearly fourfold increase in the static content capabilities of the Web server.

Apache Recommendations

The following recommendations were taken from the Apache website at: <http://www.apache.org/docs/misc/perf-tuning.html>. They are post-compile optimizations that Compaq implemented for this performance testing.

- **Turning off Hostname lookups:** Edit the server's `httpd.conf` file:

```
HostnameLookups off
<Files ~ "\.(html|cgi)$">
    HostnameLookups on
</Files>
```


- **Configuring the `httpd.conf` file to allow overrides:** Edit the server's `httpd.conf` file:

```
DocumentRoot /usr/local/httpd/htdocs
<Directory />
    AllowOverride none
</Directory>
```

- **Disabling symbolic links:** Edit the server's `httpd.conf` file:

```
DocumentRoot /usr/local/httpd/htdocs
<Directory />
    Options -FollowSymLinks
</Directory>
<Directory /cgi-bin>
    Options -FollowSymLinks
</Directory>
```

- **Turning off default document negotiation:** Edit the server's `httpd.conf` file:

```
DirectoryIndex index.html
```

- **Creating the process:** To prevent the start-up costs of the `httpd` daemon from being measured, configure the following in `httpd.conf`:
 - Set the `MinSpareServers` value to 50.
 - Set the `MaxSpareServers` value to 150.
 - Set the `MaxRequestsPerChild` value to 3,000,000.
 - Set the `MaxClient` value to 256.
 - For the benchmark testing proper, the `StartServers` parameter was set to 150 to minimize startup lag times.

Updates to Default Settings

Compaq modified these default settings:

- **Logging:** Since most Web servers maintain logs, Compaq enabled logging at a low level and mounted the directory `/var/log` (into which Compaq wrote log files `httpd.access_log` and `httpd.error_log`) on its own partition on a disk separate from the main OS disk.
- **`noatime` option:** Since Web server log files provide access times for files, Compaq mounted the Apache document root with the `noatime` option on a separate partition within the same disk as the main OS disk. However, the results of the performance testing showed that this option had a minimal impact on performance.

- **Optimizing the Linux 2.2.4 kernel:** Compaq rebuilt the Linux 2.2.4 kernel and added some optimizations including optimizations for 6x86-class machines and the removal of several kernel options not necessary for the *ProLiant* 1850R server. The `.config` file that determined the Compaq kernel build accompanies this document as sample code⁷.

Test Results

This section discusses the results of the Compaq testing and identifies key factors that influence Web server performance with static or CGI content. Compaq provides recommendations that can allow customers to improve Web server performance.

Static Performance: Network Bandwidth and RAM

Web servers with a high percentage of static content (HTML) are more likely to experience network bandwidth as a gating factor rather than CPU utilization limits. This is especially significant since many Web servers have no more than a single T1 line to provide network access. The results of the Compaq testing indicate that the Apache HTTP Server running under Linux on Compaq hardware is capable of handling more network throughput than that offered by several T1 lines.

Another limiting factor for a server with primarily static content is the amount of RAM available to the server. This becomes more critical as bandwidth increases. Figure 2 shows that a server with just 128 MB of RAM serving up a 512 MB workload set (where there is not enough RAM to cache all the content) has a total network throughput greater than that provided by 5 T1 lines. The amount of RAM does not become a factor until this throughput is exceeded.

⁷Use the appropriate link on the *ActiveAnswers* page from which this document was downloaded.

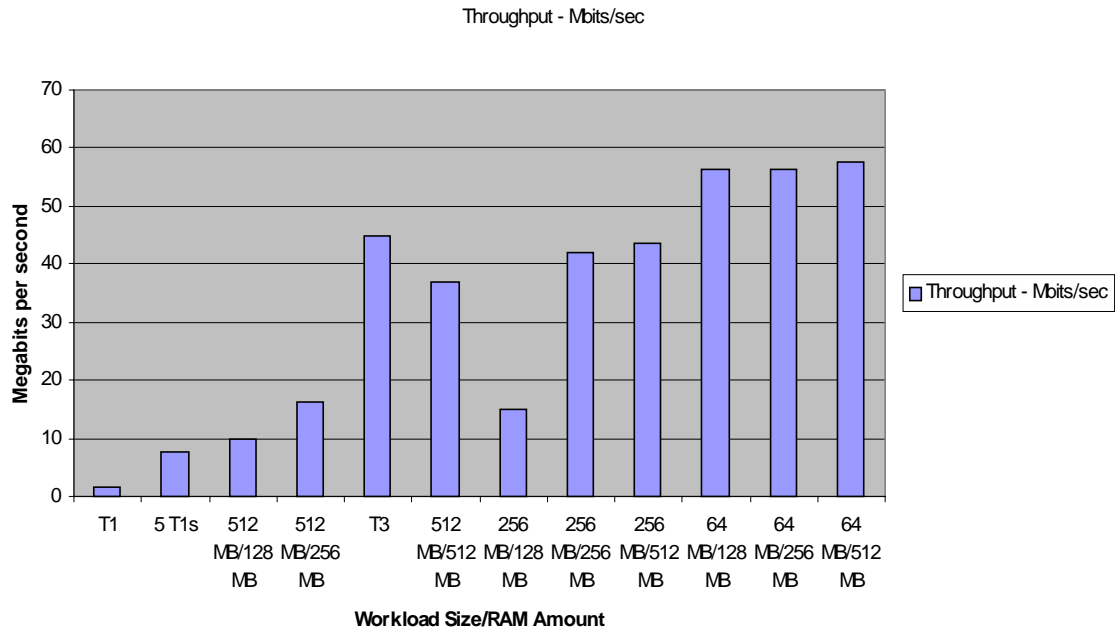


Figure 2. Network throughput and RAM utilization

Once the total throughput exceeds the capabilities of between 5 and 7 T1 lines, then the amount of RAM available to the server becomes the primary factor influencing Web server performance. However, Compaq testing indicated that a single server can accommodate the throughput of a T3 line if enough RAM is installed. This is apparent both from the throughput shown in figure 2 and also from the number of requests per second serviced shown in figure 3. Both figures show that, as the workload size exceeds the capabilities of the RAM available to the system, server performance suffers.

This is even clearer in figure 4, which shows the CGI and static results side-by-side. Static web serving performance decreases to almost the level of the simple CGI performance when there is insufficient RAM. As the available RAM increases, the static performance increases proportionally to the point that significantly more static requests can be processed than CGI requests.

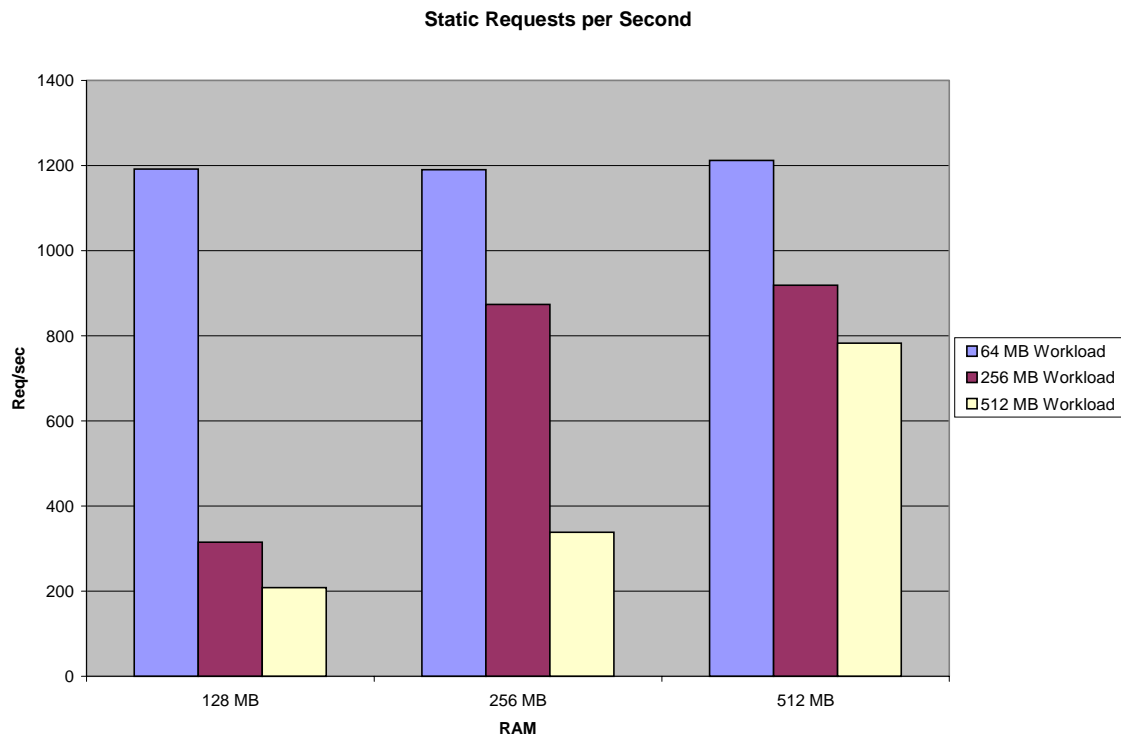


Figure 3. Static requests per second with increasing RAM

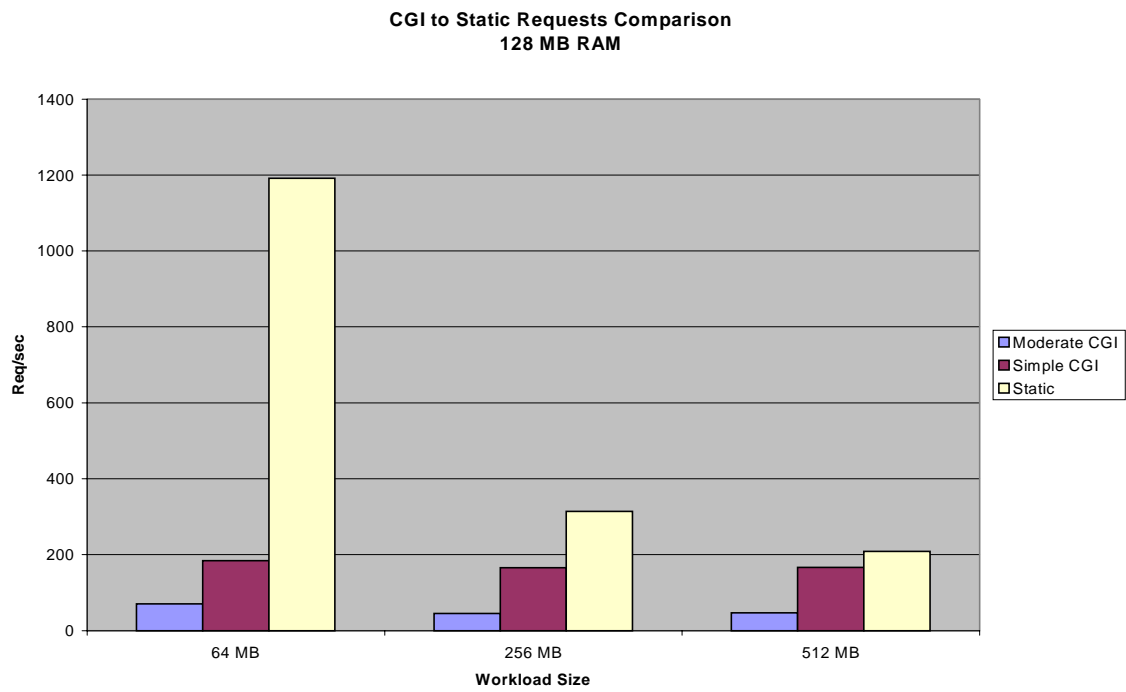


Figure 4. Static requests per second and CGI requests per second, 128MB RAM

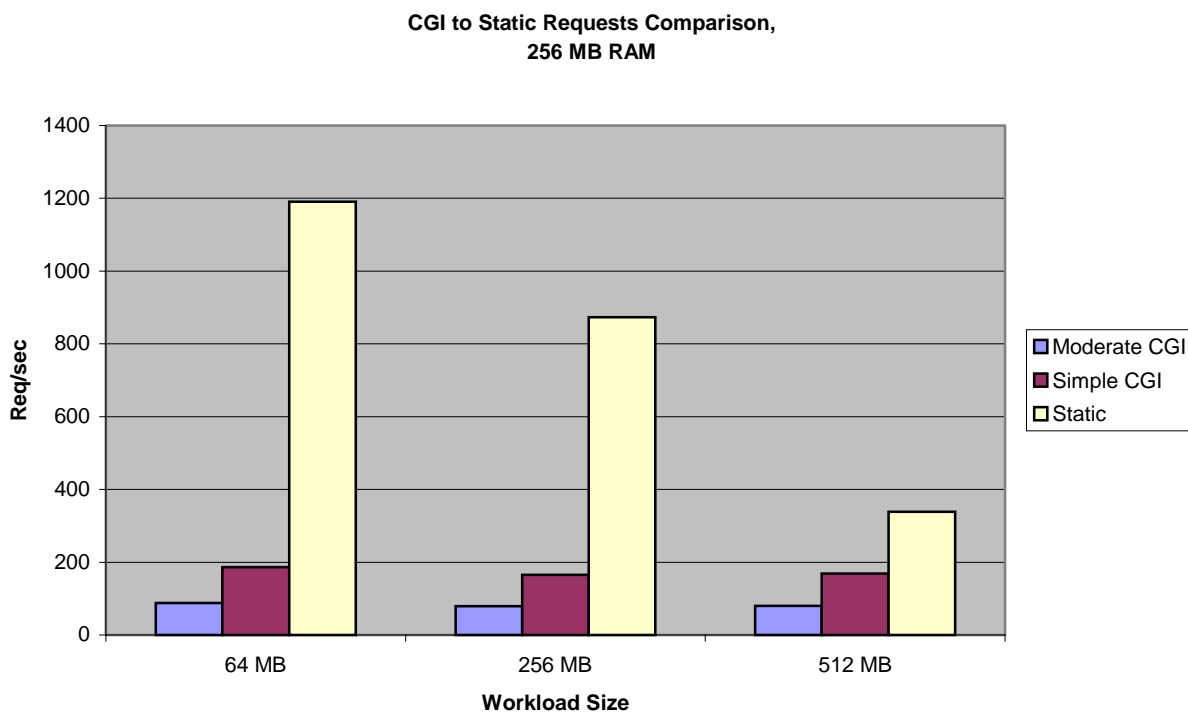


Figure 5. Static requests per second and CGI requests per second, 256MB RAM

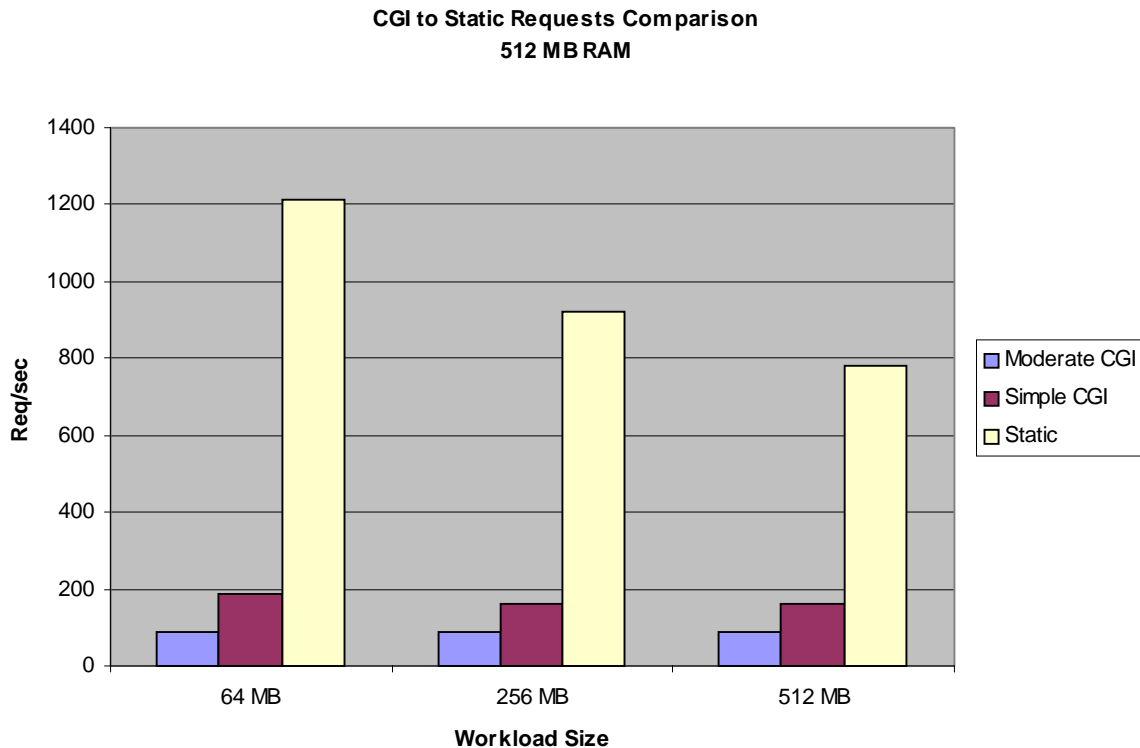


Figure 6. Static requests per second and CGI requests per second, 512MB RAM

Figures 4, 5, and 6 demonstrate clearly that, as the amount of RAM increases, the server's capability to serve up static pages increases correspondingly. For Web servers that primarily serve up static pages, the customer can make a relatively small investment in RAM in order to greatly increase performance. However, this investment may not be warranted if the network bandwidth to the server is so low that RAM is not a performance limiter.

Static Performance: Workload Size

The relationship between the workload size and the amount of RAM also affects Web server performance. If the workload size greatly exceeds the amount of RAM available to the system, the performance of a static-only Web serving system degrades to a similar level to that of a CGI-only system. In such cases, the customer can purchase additional RAM to improve static-only performance.

However, since most websites serve up only a fraction of their total content in response to typical requests, the customer should analyze the Web server log files to identify the most common content size. In the Compaq test environment, every file in a given workload set had more or less the same probability of being served: in the real world, the total content at many websites can be 1GB or more⁸ but the system only serves from a 256MB subset 70% - 80% of the time. The customer should identify the size of the subset that is most frequently served and allocate enough RAM to accommodate that subset.

Static Performance: Recommendations

Based on this testing, Compaq recommends that the customer should determine the maximum throughput that the network can present to the Web server. For a low throughput, the customer should deploy a machine configured with a minimal amount of RAM (128MB); for a higher throughput (more than 5 T1 lines), the user should configure additional RAM to cache the subset of files that are most frequently served.

CGI Performance: Overhead

Figures 4, 5, and 6 demonstrate little correlation between the amount of RAM available to the Web server and the performance of CGI applications. The number of requests served per second is virtually the same regardless of the amount of RAM available.

This can be attributed to the manner in which CGI programs are executed:

1. The CGI program is found on disk (or in the file system cache) and launched.
2. The CGI program executes its functionality.
3. The CGI program process is terminated.

The test results suggest that the overhead required to complete these steps is so great that the available RAM cannot be effectively utilized. Since this is mostly execution overhead, the processor must work harder to accommodate it: this, in turn, suggests that increasing the processor capacity is the best way to improve the performance of CGI applications.

⁸ More than Compaq used for this testing.

CGI Performance: Additional Processor

Compaq conducted tests to compare the performance of single and dual processor systems. The results indicated that, while the performance improvements were minimal, the dual processor system performed better than the single processor system in all tests. In particular, testing with moderate CGI content demonstrated a more significant performance increase than testing with simple CGI or static content.

Figure 7 shows the number of requests per second for single and dual processor tests; figure 8 shows the percentage increase from one test to the other using the single processor test as the denominator.

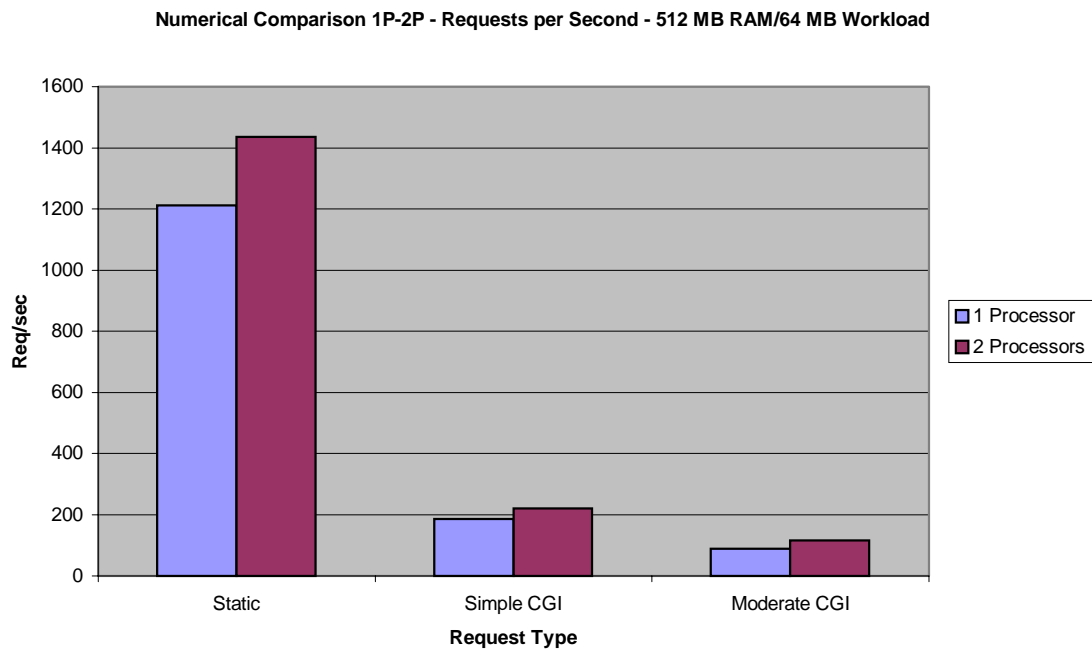


Figure 7. Numerical comparison between single processor and dual processor performance

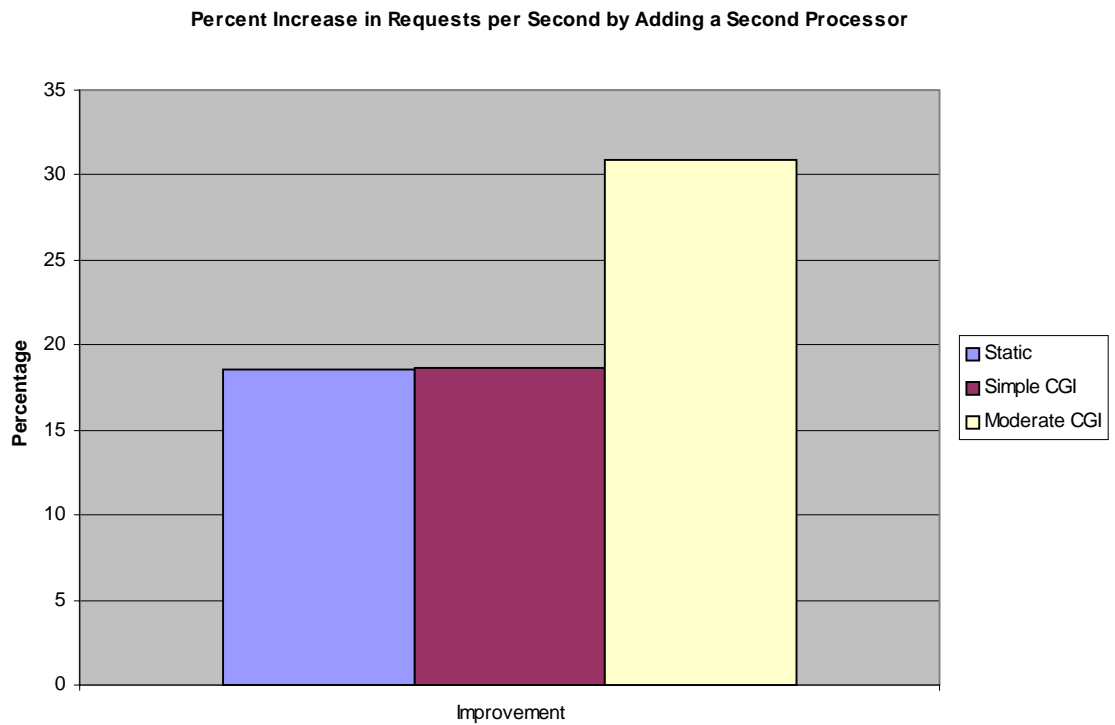


Figure 8. Percentage performance improvement by adding a second processor

CGI Performance: Recommendations

- **Faster processor:** For a CGI-intensive Web server application, Compaq recommends using the fastest processor that the customer can afford.
- **Additional processor:** Although the performance increase will not be dramatic, the customer should consider adding a second processor to increase the overall performance of the Web server. If the site serves more intensive CGI requests, the customer can expect a greater performance increase when upgrading from a single processor to dual processors than if the site serves mostly static or simple CGI requests.

Appendix: Source Code for Moderate CGI Program

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int HexToDec( char c )
{
    switch( c )
    {
        case '0':
            return 0;
        case '1':
            return 1;
        case '2':
            return 2;
        case '3':
            return 3;
        case '4':
            return 4;
        case '5':
            return 5;
        case '6':
            return 6;
        case '7':
            return 7;
        case '8':
            return 8;
        case '9':
            return 9;
        case 'a' : case 'A' :
            return 10;
        case 'b' : case 'B' :
            return 11;
        case 'c' : case 'C' :
            return 12;
        case 'd' : case 'D' :
            return 13;
        case 'e' : case 'E' :
            return 14;
        case 'f' : case 'F' :
            return 15;
        default:
            return 0; // uh-oh
    }

    return 0; // uh-oh
}
```

```

void dump( char *p, FILE *fp, int len )
{
    int iAsciiEscape;
    int iAsciiValue;
    int i;
    char *pOut;
    int j;

    pOut = (char *)calloc( len, sizeof(char) );

    iAsciiEscape = 0;
    j = 0;
    for( i = 0; i < strlen(p); i++ )
    {
        if( iAsciiEscape )
        {
            if( iAsciiEscape == 1 ) // first digit
            {
                iAsciiValue = HexToDec( p[i] );
                iAsciiValue *= 16;
                iAsciiEscape++;
            }
            else // second digit
            {
                iAsciiEscape = 0;
                iAsciiValue += HexToDec( p[i] );
                pOut[j++] = iAsciiValue;
            }
        }
        else
        {
            if( p[i] == '+' )
            {
                pOut[j++] = ' ';
            }
            else if( p[i] == '&' ) // separator
            {
                pOut[j++] = '\n';
            }
            else if( p[i] == '%' ) // begin ASCII escape sequence
            {
                iAsciiEscape = 1;
                continue;
            }
            else
            {
                pOut[j++] = p[i];
            }
        }
    }

    pOut[j] = '\0';
    fprintf( stdout, "%s", pOut );
    fprintf( fp, "%s", pOut );
    free( pOut );
}

```

```
main()
{
    pid_t id;
    int u;
    FILE *fp;
    char *p;
    char fn[64];
    char buffer[2048];

    id = getpid();
    u = (int) id;
    sprintf( fn, "./File%d.dat", u );
    fp = fopen(fn, "a+");
    if( !fp )// could not open file
    {
        printf( "Server-Type: Apache 1.3.4\n\n<html>\n
<body><h1>Could not open file ./File.dat</h1></body></html>\n
\r\n");
        exit(-1);
    }

    printf( "Server-Type: Apache 1.3.4\n\n<html>\n
<body><P><strong>");
    p = getenv("QUERY_STRING");
    fprintf( stdout, "QUERY_STRING:\n");
    fprintf( fp, "QUERY_STRING:\n");
    dump( p, fp, strlen(p) );

    fprintf( stdout, "<br>POST_DATA:\n");
    fprintf( fp, "POST_DATA:\n");
    while( (p = fgets( buffer, 2047, stdin)) != NULL )
    {
        dump( p, fp, 2048 );
    }

    printf( "</body></html>\r\n" );
    fprintf( fp,\n
"\n===== \n" );

    fclose( fp );

    return 0;
}
```