digital™

DIGITAL AlphaServer for Intranet Search

White Paper: The Effect of Memory Capacity on the Search Response Time of an Intranet Search-engine Server running in the Windows NT™ Environment

Table of Contents

Abstract

DIGITAL customers who are planning to implement intranet search solutions need to better understand the implications of memory capacity on search response time. To provide this information, engineers at DIGITAL's Computer Special Systems (CSS) group conducted search-response-time characterization tests on the DIGITAL AlphaServer™ for Intranet Search platform using 128 MB and 512 MB memory configurations on an AlphaServer 800 running Windows NT 4.0 and the AltaVista™ Search Intranet Private eXtensions software.

The study used two indexes of different sizes: Index 1 contained 70,589 web pages while Index 2 contained 129,706 web pages. In theory, the higher the memory capacity, the greater the workload that can be handled. For this particular study, though, the two indexes were both small enough to fit in 128 MB physical RAM (or nearly so). Thus there was no dramatic improvement in response time between the 128 MB and the 512 MB memory configurations except at lower loads. Scaling up to 512 MB is indicated if index size will be significantly larger than Index 2, either initially or as growth occurs.

The results of the testing will serve to help customers implement appropriately sized servers for their projected indexing requirements.

A similar study was carried out in the DIGITAL UNIX® environment using an AlphaServer 4000 system. This study is described in a white paper accessible on the World Wide Web at **http://www.digital.com/info/**.

Introduction

Any organization installing an intranet search engine should do so only after developing a plan for long-term growth. There may be a surprising surge of use as users discover the search engine and come to rely on it for an increasing number of searches.

As time goes on, legacy data may be connected to web pages. Some of this may be dynamic, but a good measure may be static, or at least static enough to be targeted by a search engine.

As the index size increases, disk utilization will increase as well. Not so obvious is the fact that an intranet's index might come to be seen as mission-critical. In this case, various disk redundancy approaches may be needed. Furthermore, the search computer itself may need to assure high availability.

As an index grows, the load of search requests can become greater than originally anticipated. At this point, disk performance can be improved by a variety of means including an efficient compression scheme. This compression should not, however, unduly impact the time required for a search.

For a user, the elapsed search time (search request latency) is critical. If routine searches take too long — for any reason — the usefulness of the search engine is compromised. The search engine platform may have to be upgraded if performance suffers. This could include adding more memory, processors, disk controllers, or other resources. Or the bottleneck affecting search-request latency may be network bandwidth available to the search engine.

A newly installed intranet search engine might need only an Ethernet connection at first. As usage rises, though, it would be wise to have a plan in place for upgrading bandwidth in a number of ways. A faster controller might help or a dedicated port on a switch might be necessary. One can quickly run out of simple solutions as the source of the bottleneck moves "outward" into the entire network infrastructure. Local or enterprise routers, or WAN connections into the building, might be overtaxed. It's even possible that another building's network might begin to see noticeably heavier use as the occupants discover the new search engine.

It is impossible to anticipate every possible network and client/server configuration. But to help quantify the performance characteristics of intranet search implementations for planning purposes, it is useful to have a baseline characterization of some component of the overall system. The most obvious and quantifiable component is search-engine server memory, the subject of this study.

The next section of this report presents "A Search Engine Primer." If you are familiar with AltaVista search engine technology, you can skip this section.

A Search Engine Primer

The AltaVista search engine consists of three components.

The data collector

The first component is the data collector, also known as a crawler, robot, or “super-spider.” The AltaVista Search Service data collector, dubbed “Scooter” by DIGITAL, is the fastest known super-spider in existence. The data collector finds web pages by following URL cross-references from one or more starting pages. One of the many things that must be done by this component is to eliminate duplications, that is, to notice when a “URLed” page has already been visited.

The indexer

The second component is the indexer. This component retrieves the entire text of all found web pages (as determined by the data collector) and indexes all words therein. Obviously, most words will have been found on more than one web page, a fact that must be considered in the design of the index.

The query interface

The third component is the *query interface*. This component encompasses the graphical user interface that most users see. It operates on the index (as built by the indexer) and supplies URLs and other information about web pages which match the criteria requested by the user.

More information is available

For a detailed explanation of the AltaVista search engine, we recommend the following book, *The AltaVista Search Revolution*, by Richard Seltzer, Eric J. Ray, and Deborah S. Ray, published by Osborne/McGraw-Hill.

To experience AltaVista on the Internet, go to **www.altavista.digital.com.**

Response-time Test

Test Platform

The test platform consisted of a single-CPU AlphaServer 800 model 5/400 MHz (Alpha 221164 CPU) running AltaVista Search Intranet Private eXtensions v1.0b SP2 software as the search engine.

The AlphaServer 800 ran BIOS v5.28. The disk drive was a reserved 2 GB RZ28M-S on a Qlogic ISP1020 PCI controller (fast wide SCSI, firmware v4001). This spindle was non-compressed NTFS, different from the system/swap spindle. The operating system was Windows NT v4.00.1381 (build 381 aka SP1). Performance was set for maximum boost for foreground applications, and to maximize throughput for networking (as opposed to file I/O). Tests were run on two memory configurations: 128 MB and 512 MB.

AltaVista Search has a feature designed to provide fairness by subnet. This feature had to be overridden for use with one subnet containing a large number of requests. In httpd/config the following settings were employed:

Sockets	100
Threads	20
SubnetSockets	100
SubnetThreads	10

Test Overview

The team built a private FDDI intranet comprising several nodes (Figure 1). The only traffic on the test intranet was that which resulted from the test trials. FDDI was selected as the network medium to minimize any bandwidth-limitation effect.

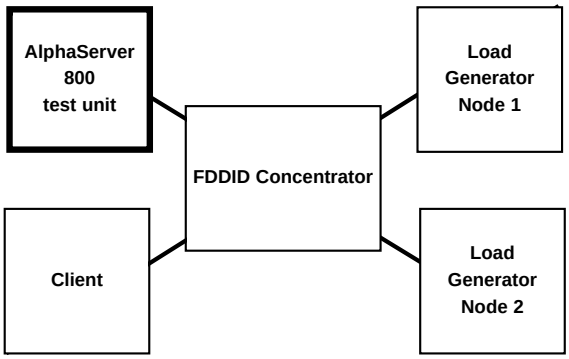


Figure 1: Test network topology

The characterization testing set out to determine response times for the AltaVista intranet product under varying workloads and memory configurations. Workloads are defined by the number and frequency of browser-based search requests on the test index.

This testing did not require a large number of computers because varying workloads were simulated by a small program with two main functions. First, to generate a search request and, second, to record the elapsed time from request initiation to reply reception. The program does not utilize any other resources, such as an X display.

Load-generator Nodes and the Phrase Pool

We created a “phrase pool” consisting of a large number of search-phrase lists. For each trial, load-generator nodes created many separate executing shells, each emitting a “stream” of search requests from the phrase pool in such a way that no search phrase is ever re-used during the trial. This eliminates the possibility of the server anticipating repeat requests.

Index

The index itself was acquired from within the Digital Equipment Corporation intranet, and consisted of a mix of pages, mostly technical in nature, of varying sizes. Two different sized indices were created for use in different scenarios.

- Index 1 contained 70,589 web pages
- Index 2 contained 129,706 web pages

Server

The AltaVista Search engine was run on the AlphaServer 800 server. The executables and, most important, the entire index, were placed on a separate disk drive to minimize the effect of other system activity such as swapping. The system was “tuned” for the task at hand, overriding as required those default values which are more appropriate for other general purposes.

At the start of each trial, the target disk drive was deleted and restored from a save area, thus assuring that all files were in the same disk locations. For any one scenario, the available system memory was varied over multiple trials for each product configuration.

Load Generators

Two nodes were used to generate a large "background" load, i.e., many search requests per time unit. For each scenario, a number of trials were run, varying the number of request streams starting from a nominal low point and ending when the request load pushed the response time into an "unacceptable" range of many seconds.

Client

A separate client node was lightly loaded to minimize any interference between the load generation and the response-time measurement of a sample request stream.

Response time recording

A stream of 100 non-recurring search phrases was sent to the server while the load generators were sending their background load. The response time for these 100 requests was recorded automatically. The actual timing of the start of each request was randomized by the method of waiting for an operator to press a key. This approach was independently checked for reliability, and was found to be reproducible within a few percent.

Data Analysis

For each trial, the actual generated load was computed automatically as the number of TCP connections made during a three-minute time period. The values from each of the two load generators were added to form the total load measured in requests per hour.

For that trial, the 100 client response times were subject to two calculations. First, a simple arithmetic mean (average) was computed. Then the 100 response times were sorted by value and the 11th slowest time captured. 90 percent of the requests were handled within this time, while 10% required longer. As a further check of this method, the 10th and 9th slowest times were also recorded. In actual trials, no anomalies (such as a big jump among these three numbers) were seen.

One such trial, for a given scenario, would thus produce an average time and a "90%" time for a particular load (in hits per hour). Repeating the trial with a different number of search-request streams on the load generators would produce another such data point. Finally, all of the data points for each given scenario were plotted.

Limitations: Screen-rendering time

Note that the elapsed time measured by the test program does not include one important aspect of real browser operation, namely the time to "paint" the screen, i.e., render the HTML. Screen-rendering time is very dependent on client computer characteristics including CPU, memory, and video resolution and speed so we decided to exclude this time. In a real network of thousands of users, each browser would be running on a separate computer and this screen-rendering time would be spread over all these nodes. The extra time impact would only affect each user once.

Limitations: Network bandwidth

Note also that bandwidth limitations of various network equipment, including the chosen media, server controllers, and especially dial-up modems if employed, will all serve to mask the actual underlying performance of the search engine. Modems are usually employed as only one final link to the user, in which case that delay would only affect each user once. However, where a WAN link is insufficiently sized for the intended traffic, a significant global impact will occur.

Response-time Test Results

The "Wall"

All of the graphs generated by this study have a characteristic shape, relatively flat and only slowly increasing with greater load, until a point at which the response time sharply increases with only small increases in the load. This point is termed the "wall" of maximum load (for a given test scenario).

X-axis Load Indirectly Controlled

An oddity of the data points is that the X-axis, described as load or hits/hour is, in fact, not directly controlled by the test bed. Instead, each trial can *attempt* a particular load, by starting a certain number of request streams on each load generator. However, the method of running request streams doesn't allow for the variable delay which occurs as a result of the server's load (due to *other* streams). Therefore, the actual load must be measured during the test trial.

Y-axis Two Values: Average and 90%

For each actual measured load in a trial series, two Y-axis response time values are plotted: the arithmetic mean or average, and the "90%" threshold. The former is always lower, i.e., represents a faster response time.

Backward Slope at High Attempted Loads

Another oddity of the graphs is that most show a negative slope at the high end of the plotted line. This occurs because, after a certain point, increasing the attempted load (by a larger number of request streams) actually decreases the actual load.

Noise at High Attempted Loads

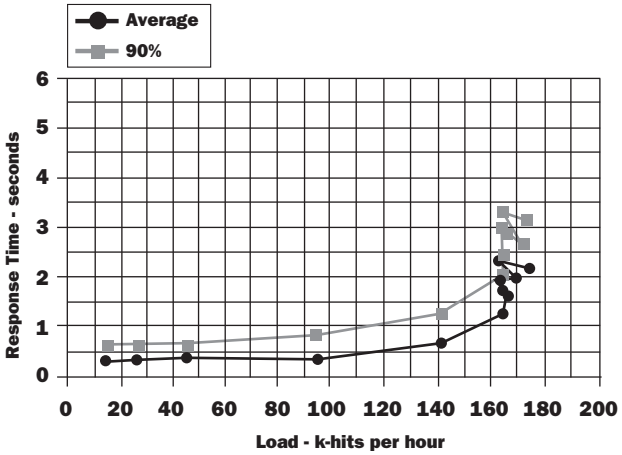
Most graphs show some "noise" at very high loads, i.e., after the point at which the load approaches the wall of maximum load. This is probably the result of miscellaneous effects which are hard to quantify (but fortunately are small). Possibilities include the exact coincidence of requests and other system activities such as swapping and various time-outs.

Small Index Test

This is actually a decent sized index containing 70,589 indexed web pages.

128 MB Memory Size: Small Index

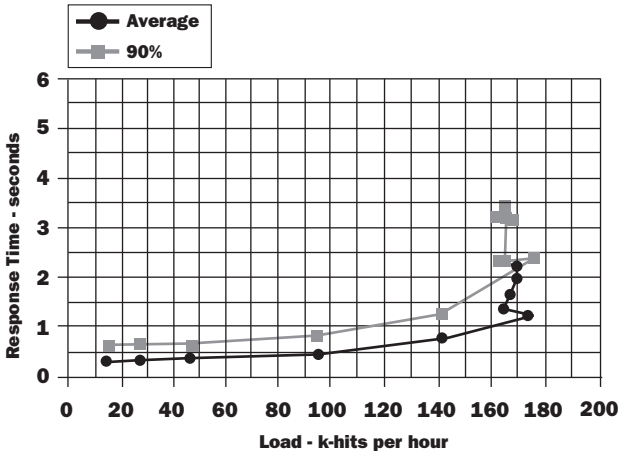
Graph 1 shows that this memory size with this index size has good response time (about .5 second) up to the wall of maximum load at about 170K hits/hour (about 47 per second).



Graph 1: AltaVista Search Server AlphaServer 800 (128 MB memory) 70k-pages index on WNT 4.0

512 MB Memory Size: Small Index

Graph 2 shows that this memory size with this index size has no improvement over the smaller memory size. The wall of maximum load is the same as for 128 MB in Graph 1. Some noise is seen at the highest loads.



Graph 2: AltaVista Search Server AlphaServer 800 (512 MB memory) 70k-pages index on WNT 4.0

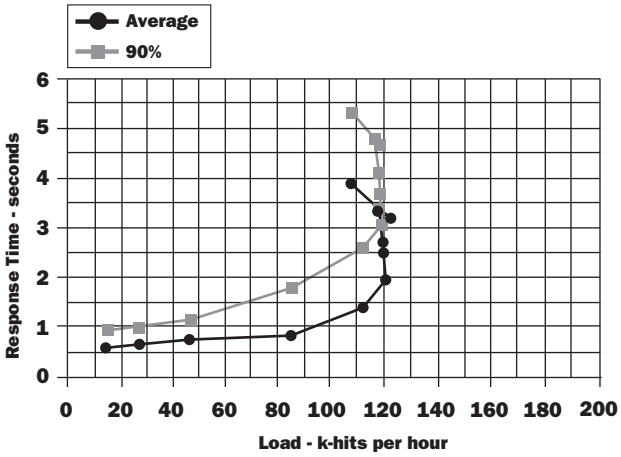
Medium Index Test

This medium sized index describes 129,706 web pages.

128 MB Memory Size: Medium Index

Graph 3 shows that this memory size with this index suffers even at low load, when compared to the smaller index in Graphs 1 and 2. The wall of maximum load is much lower.

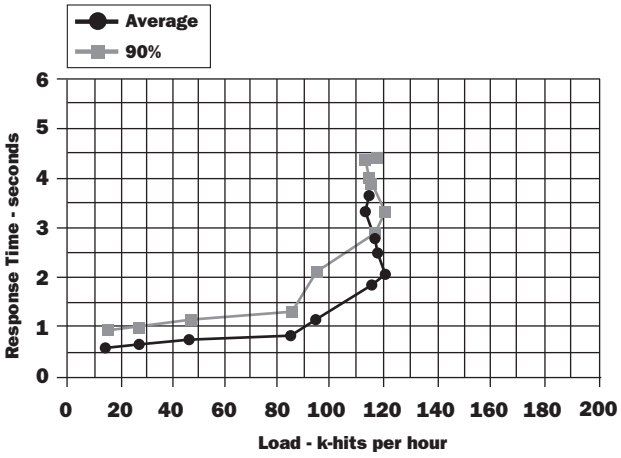
In this case, the maximum load is about 120K hits/hour (about 33 per second). The backward slope at higher loads is very apparent.



Graph 3: AltaVista Search Server AlphaServer 800 (128 MB memory) 129k-pages index on WNT 4.0

512 MB Memory Size: Medium Index

Graph 4 shows that this memory size with this index size has no improvement over the smaller memory size. The wall of maximum load is the same as for 128 MB in Graph 3. Some noise is seen at the highest loads.



Graph 4: AltaVista Search Server AlphaServer 800 (512 MB memory) 129k-pages index on WNT 4

Response-Time Test Conclusions

This characterization testing set out to determine response times for the AltaVista intranet search engine under varying workloads and memory configurations. Workloads are defined by the number and frequency of browser-based search requests on the test index.

As we have seen, all of the graphs generated by this study have a characteristic shape: relatively flat and only slowly increasing with greater load, until a point at which the response time sharply increases with only small increases in the load, indicated by the “knee” in the graph. This point is termed the wall of maximum load (for a given test scenario). Response time performance for any given index size and memory configuration eventually hits the wall.

In theory, the higher the memory capacity, the greater the workload that can be handled. For this particular study, though, the two indexes were both small enough to fit in 128 MB physical RAM (or nearly so). Thus there was no dramatic improvement in response time between the 128 MB and the 512 MB memory configurations except at lower loads reaching half the wall. Scaling up to 512 MB is strongly indicated if index size will be significantly larger than Index 2, either initially or as growth occurs.

CPU utilization was also measured during some trials. We determined that CPU utilization approached 100% at the maximum supported load for each memory configuration, *regardless of index size*. This implies that the characterizations for this computer are appropriate and indicates that one would probably not achieve appreciably better search processing by adding memory or *any other* resource.