Advantage

Database Server

scalability
technical brief

November 2005

Copyright and Trademarks

Table of Contents

Introduction

While Advantage Database Server is ideal for client-server applications deployed to small and medium-sized businesses, it can also run high-traffic web sites that support hundreds of customers. With Advantage, web developers have the flexibility to combine powerful SQL access with the performance and control of navigational commands providing optimized data access methods unmatched by any other database.

This technical brief provides an overview of an internal scalability test using Advantage Database Server 8.0. A bookstore web application was developed similar to the Nile Bookstore web application used in a 2002 eWeek database benchmark. Trademark restrictions prevent publication of the script, but the benchmark can be re-created by downloading the code and data from Server Databases Clash (www.eweek.com/article2/0,1895,1646917,00.asp).

The site performed the same basic functions of a full-featured e-commerce site: browse books by subject, search books by subject, author or title; allow users to add books to a shopping cart; and allow users to checkout (purchase) the books, calculate sales total, and submit orders.

This test used larger-than-average datasets for a typical application of this type. A load was applied to the dataset using Microsoft Application Center Test (ACT), which has the ability to simulate simultaneous browser connections. It executes a script from the client browser. In our testing, the script performed the following functions:

1. Access the main page and display product specials
2. Select a link to browse books by subject
3. At random, perform six browse operations
4. Perform two search operations
5. Log into the site
6. Add a book to the shopping cart
7. Select checkout
8. Submit final order

Specifications

Tables

Customer	50,000 records (16,117 KB)
Orders	variable
Order Details	variable
Products	1,000,000 records (179,690 KB)
Session	variable
SessionData	variable
ShoppingCart	variable
Subjects	21 records (2 KB)

Server Machine

HP ProLiant ML370

Two Intel Xeon DP Processors 3.6GHz

1GB RAM

Windows 2003

IIS 6.0

Client Machine

Compaq nc8000

Intel Pentium 1.6 GHz

512 MB RAM

Windows XP SP2

Development Tools

Visual Studio 2003

C#

Advantage .Net Data Provider 8.0

Advantage Database Server 8.0

Scalability Test Results

Web site performance was measured using requests per second (RPS) and time to last byte (TTLB). For additional information on performance tuning and scalability, see Improving .NET Application Performance and Scalability on Microsoft's MSDN web site (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenet.asp).

Test runs were recorded using 100, 200, 300, 400, and 500 concurrent users without delays enabled. This is an abnormally heavy load to stress-test the server. In typical web site behavior, it is unusual for 100 users to be concurrently performing database operations. In general, the majority of those users are connecting and browsing without actually performing data access.

Definitions

Requests Per Second (RPS): The number of requests sent per second, not including requests that are sent multiple times.

Time To Last Byte (TTLB): The Time to Last Byte measures the time between sending the request to the server and receiving the last byte of the response.

Time To Last Byte

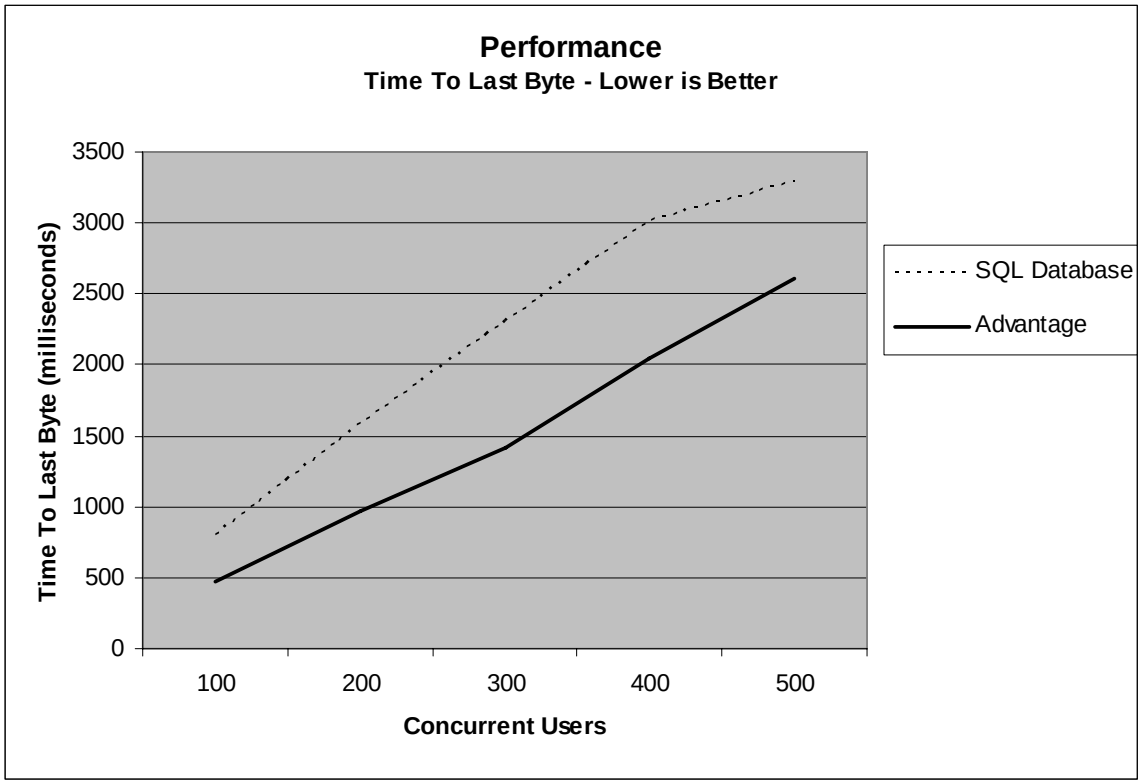

Figure 1

Time to Last Byte (TTLB) is the best measurement of web site responsiveness under a heavy load. In ***Figure 1***, Advantage scales to 500 users with browser responses less than 500 milliseconds at 100 users, and less than three seconds with 500 users. When measuring TTLB, a faster response rate is better. To add perspective, ***Figure 1*** includes results from a comparable SQL-based database. (Note: Most major database vendors prohibit direct benchmark comparisons in their licensing agreements. The comparisons used in this white paper are intended to provide a frame of reference for scalability purposes.) At 500 users, the response is more than three seconds with the non-Advantage server.

Requests Per Second

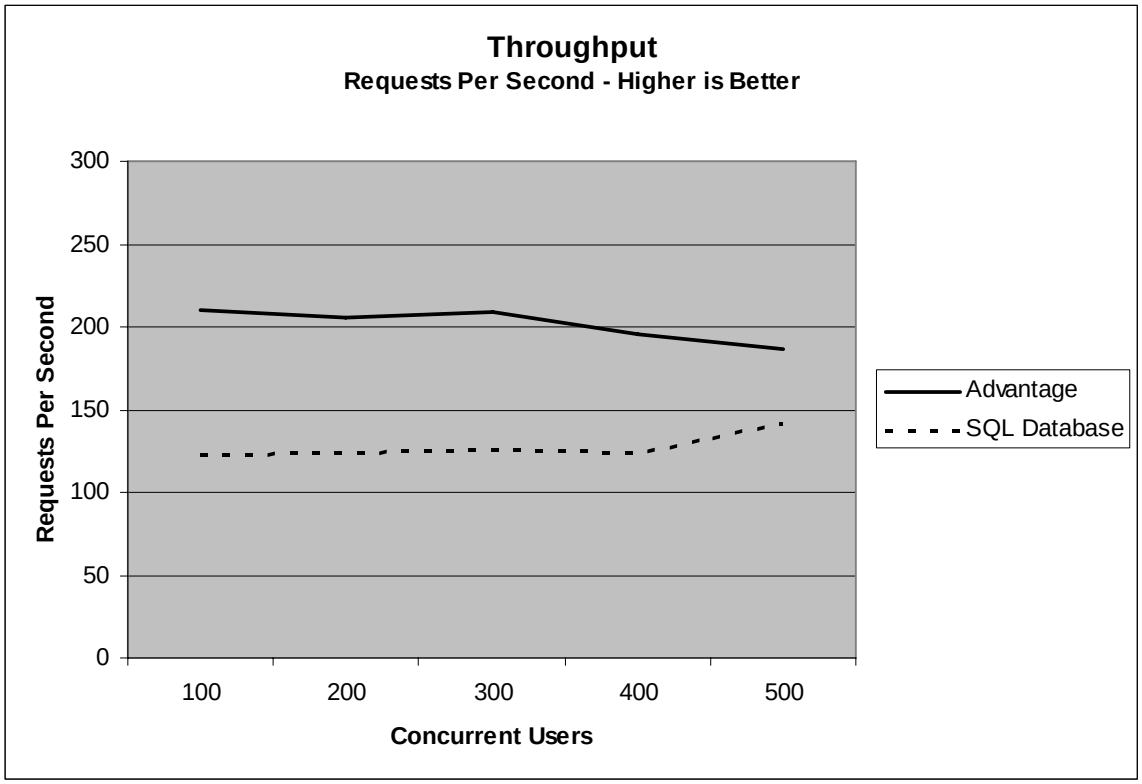

Figure 2

Requests Per Second (RPS) is the best measurement of web site throughput. In ***Figure 2***, Advantage scales to 500 users while maintaining a fairly consistent RPS. The web page running against Advantage can handle an average of 187 RPS even as the user count reaches 500. In contrast, the SQL-based database executed 141 RPS with 500 users.

Web-based scalability tests are strongly influenced by hardware constraints. The goal of this scalability test was to provide a reasonable expectation of database performance with a common hardware platform. Throughput and responsiveness could have been improved with a higher performance and a higher cost hardware configuration.

Optimization

The following are Advantage-specific optimization techniques that should be considered when developing Advantage-based web sites:

1. Use Advantage Database Server version 8.0.

   Version 8.0 includes many enhancements that will significantly improve web site performance including aggressive index and static cursor caching, improved transaction processing performance, faster SQL parsing, faster filtered record count operations, optimized queries, and server-to-client direct communication. ***Figure 3*** demonstrates the performance gains between version 7.1 and 8.0 using the web site scalability application.

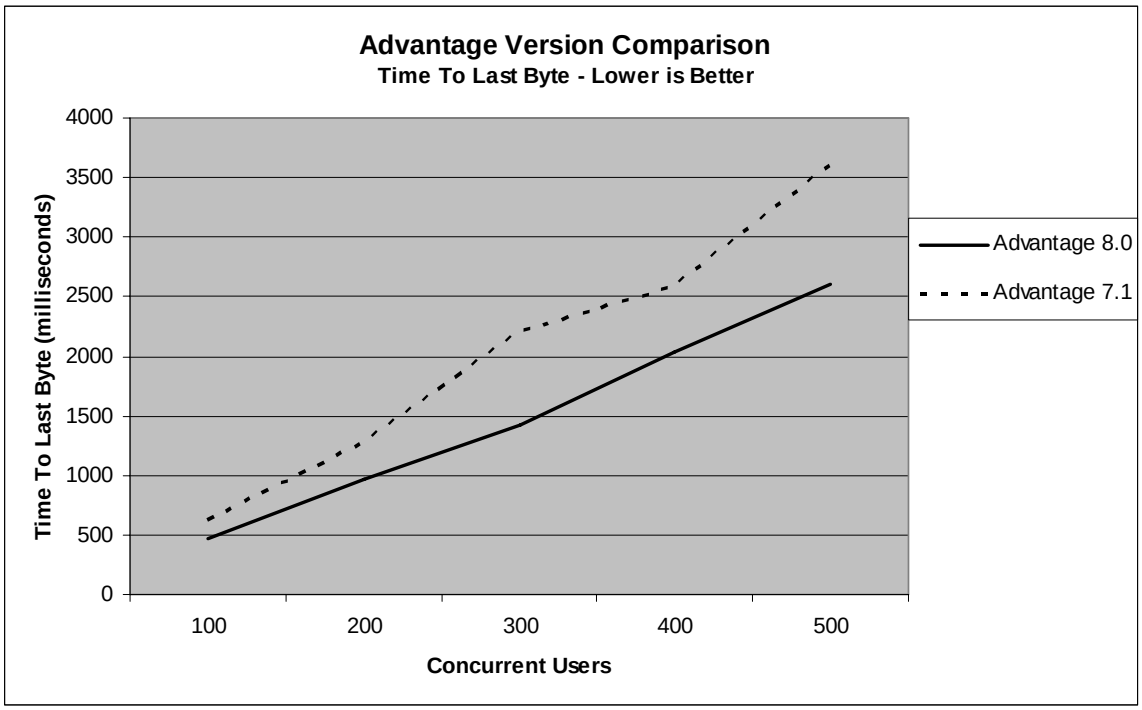

Figure 3

2. Use live cursors when possible.

   With the Advantage Database Server, a live cursor is generated when an SQL statement maps directly to a single table. A static cursor joins one or more tables or uses certain scalar functions. For instance, “SELECT * FROM tableA WHERE lastname = ‘Smith’” generates a live cursor and “SELECT * FROM tableA, tableB WHERE a.pk = b.pk” generates a static cursor. Generally, a live cursor that is fully optimized with appropriate indexing will execute quickly, even with hundreds of users. Internally, Advantage sets an Advantage Optimized Filter. Internal testing has shown that Advantage can execute live cursors and optimized filters much faster than other competing databases. In this scalability test, replacing a static cursor JOIN with two live cursors improved performance as the user load increased.

3. Use the AdsExtendedReader.

   AdsExtendedReader is derived from AdsDataReader and offers an extended feature set that includes advanced table navigation, indexed searches, scopes, filters, locks, and direct data manipulation. By using the AdsExtendedReader.GetRecordCount method and the AdsExtendedReader.LastAutoInc property, multi-user performance increased substantially.

4. Manage connections.

   Specify the port in the connection string. If the IP address or server name and port are included in the path, the discovery process is skipped and the client connects directly to the database server with the given IP address. This is faster and avoids potential connection issues.

Multiplatform Support

Although this scalability test was performed exclusively on the Microsoft Windows operating system, Advantage supports both Linux and Netware servers. This site was developed using Visual Studio 2003 and C#, however, Advantage has a native PHP driver, JDBC driver, OLE DB provider, and ODBC driver. With these tools, the web site could have been developed with JAVA, ColdFusion, ASP, or many other development tools.

Conclusion

By using the AdsExtendedReader, Advantage Optimized Filters, and Advantage Database Server 8.0 enhancements, Advantage provides a low-cost, reliable, and easy-to-use solution for web development. Internal scalability tests demonstrate robust web site responsiveness and throughput up to 500 concurrent users.

Notes

SQL Statements

The following SQL statements were automatically collected from the Advantage Database Server during a test run via the SQL query logging that was introduced in version 8.0.

SELECT products.bookid, products.booktitle, products.author FROM products WHERE products.subjectid = 21

SELECT subjects.subjectid, subjects.subjectname FROM subjects ORDER by subjects.subjectid

SELECT bookid, booktitle, author, price, retail FROM products WHERE products.subjectid = 9

SELECT bookid, booktitle, author, price, retail FROM products WHERE products.subjectid = 1

SELECT subjects.subjectid, subjects.subjectname FROM subjects ORDER by subjects.subjectid

SELECT bookid, booktitle, author, price, retail FROM products WHERE products.subjectid = 3

SELECT bookid, booktitle, author, price, retail FROM products WHERE products.subjectid = 10

SELECT subjects.subjectid, subjects.subjectname FROM subjects ORDER by subjects.subjectid

SELECT bookid, booktitle, author, price, retail FROM products WHERE products.subjectid = 11

SELECT subjects.subjectid, subjects.subjectname FROM subjects ORDER by subjects.subjectid

SELECT bookid, booktitle, author, price, retail FROM products WHERE products.subjectid = 9

SELECT subjects.subjectid, subjects.subjectname FROM subjects ORDER by subjects.subjectid

SELECT bookid, booktitle, author, price, retail FROM products WHERE products.subjectid = 9

SELECT subjects.subjectid, subjects.subjectname FROM subjects ORDER by subjects.subjectid

SELECT bookid, booktitle, author, price, retail FROM products WHERE products.subjectid = 3

SELECT subjects.subjectid, subjects.subjectname FROM subjects ORDER by subjects.subjectid

SELECT bookid, booktitle, author, price, retail FROM products WHERE products.subjectid = 14

SELECT subjectid FROM subjects WHERE subjectname = 'Reference'

SELECT bookid, booktitle, author, price, retail FROM products WHERE SubjectID = 15

SELECT subjectid FROM subjects WHERE subjectname = 'Reference'

SELECT bookid, booktitle, author, price, retail FROM products WHERE SubjectID = 15

SELECT customerid, firstname, lastname, address1, address2, city, state, zip, email, phone, creditcard, creditcardexpiration, username FROM customer WHERE username = 'User36888' and password = 'password'

SELECT customerid, bookid, quantity FROM shoppingcart WHERE customerid = 36888 and bookid = 49

INSERT into shoppingcart (CustomerID, BookID, Quantity) VALUES (36888, 49, 1)

SELECT products.booktitle, products.author, products.price, shoppingcart.quantity FROM products, shoppingcart WHERE shoppingcart.bookid = products.bookid and shoppingcart.customerid = 36888

SELECT sum(quantity) as totalquantityordered FROM shoppingcart WHERE customerid = 36888

INSERT INTO orders (OrderDate, CustomerID, NetAmount, Tax, TotalAmount) VALUES ('2005-08-30 09:55:06', 36888, 20, 1, 21)

INSERT INTO orderdetails (Orderid, bookid, quantity) SELECT 7454, bookid, quantity FROM shoppingcart WHERE customerid = 36888

DELETE FROM shoppingcart WHERE customerid = 36888

References

Dyck , "Server Databases Clash", *ZDnet Eweek*, on-line edition, 25 February 2002, available at http://www.eweek.com/article2/0,1895,1646917,00.asp (11 October 2005).

Meier, Vasireddy, Babbar, Mackman “Improving .NET Application Performance”, Microsoft Corporation (2004), available at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenet.asp (11 October 2005).