Microsoft® .NET Framework 4.0 vs. IBM WebSphere® 7 StockTrader Benchmark Report

7/2/2010

This document supports the release of Windows Server® 2008 R2 and the Microsoft .NET Framework 4.0.

The information contained in this document represents the current view of Microsoft Corp. on the issues disclosed as of the date of publication. Because Microsoft must respond to changing market conditions, this document should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Microsoft grants you the right to reproduce this guide, in whole or in part.

Microsoft may have patents, patent applications, trademarks, copyrights or other intellectual property rights covering subject matter in this document, except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights or other intellectual property.

Microsoft, Windows Server, the Windows logo, Windows, Active Directory, Windows Vista, Visual Studio, Internet Explorer, Windows Server System, Windows NT, Windows Mobile, Windows Media, Win32, WinFX, Windows PowerShell, Hyper-V, and MSDN are trademarks of the Microsoft group of companies.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

Executive Summary

This paper presents updated benchmark results based on extensive performance and scalability testing of Microsoft .NET Framework 4.0 using the .NET StockTrader benchmark workload as well as the WSTest workload. The .NET 4.0 framework benchmark results are compared to the equivalent Java Enterprise workloads running on IBM WebSphere 7, both on a 64-bit Power6/AIX server and on Windows Server 2008 R2. Both the peak throughput rate and the relative cost of each platform are compared, so that customers can better understand both the performance and cost characteristics of the platforms tested. All tests on Windows Server were conducted on an HP BladeSystem 7000 running Windows Server 2008 R2 release (64-bit), with the final RTM release of the .NET Framework 4.0 (64-bit) which is downloadable for free on MSDN. The IBM WebSphere/Java tests were conducted using the latest IBM WebSphere 7.0 release, 64-bit. The entire benchmark kit, including .NET source code and Java source code for the StockTrader and WSTest workloads is downloadable for free at http://msdn.microsoft.com/stocktrader. This kit also includes a Capacity Planner/multi-agent benchmark tool that customers can use to perform their own testing against .NET 4.0, IBM WebSphere and Oracle WebLogic.

Platforms Tested

- **IBM Power 570 with IBM WebSphere 7 and AIX 5.3**
- **Hewlett Packard BladeSystem C7000 with IBM WebSphere 7 and Microsoft Windows Server 2008**
- **Hewlett Packard BladeSystem C7000 with Microsoft .NET and Windows Server 2008**

The table below summarizes the results of the testing, including the cost of each platform as tested. Detailed pricing data is included in Appendix A.

	IBM Power 570 with WebSphere 7 and AIX 5.3 (8 cores)	Hewlett Packard BladeSystem C7000 with WebSphere 7 and Windows Server 2008 R2	Hewlett Packard BladeSystem C7000 with Microsoft .NET 4.0 and Windows Server 2008 R2
Total Hardware + Operating System Costs	$215,728.08	$50,161.00	$50,161.00
Middle Tier Application Server Licensing Costs	$44,400.00	$37,000.00	$0.00
Total Middle Tier System Cost as Tested	**$260,128.08**	**$87,161.00**	**$50,161.00**
Trade Web Application Benchmark Sustained Peak TPS	8,016 transactions per second	11,121 transactions per second	11,020 transactions per second
Trade Middle Tier Web Service Benchmark Sustained Peak TPS	10,571 transactions per second	14,468 transactions per second	22,316 transactions per second
WSTest EchoList Test Sustained Peak TPS	10,536 transactions per second	15,872 transactions per second	23,772 transactions per second

Introduction

To perform the comparative analysis for these three application server platforms, the performance of three core workloads is examined on each platform tested:

1. **Data-driven Web application**. This comparison is based on a WebSphere 7 implementation of the IBM Trade performance benchmark application compared to the .NET StockTrader 4.0 benchmark application based on the .NET Framework 4.0.
2. **Middle-tier Web Service benchmark with middle-tier business logic and database transactions**. This comparison is based on a WebSphere 7 implementation of the IBM Trade performance benchmark application compared to the .NET StockTrader 4.0 benchmark application, testing the middle tier Web Service layer using a Web-Service based benchmark client.
3. **Web Service Benchmark based on WSTest**. This benchmark uses the WSTest 1.5 benchmark workload (as originally defined by Sun Microsystems) to focus on just raw Web Service performance, with no business logic or middle tier data access. As such, this test focuses on just the JAX-WS performance of IBM WebSphere 7 compared to the performance of the equivalent .NET 4.0 Windows Communication Foundation (WCF) Web Services.

.NET StockTrader Sample Application and Performance Kit

All sources are downloadable from MSDN as part of the .NET StockTrader sample application, available at http://msdn.microsoft.com. This download includes:

- The IBM WebSphere 7-optimized Trade benchmark application and Java source code.
- The IBM WebSphere 7-optimized WSTest 1.5 benchmark and Java source code based on JAX-WS.
- The .NET StockTrader 4.00 benchmark application and source code based on ASP.NET and WCF 4.0.
- The .NET implementation of the WSTest 1.5 benchmark and source code based on WCF 4.0.
- The Capacity Planner multi-agent benchmark tool, including source code, which enables customers to test the Web Service workloads (both for Trade and for WSTest) on their own, using the hardware/software platform of their choosing.

For the benchmark comparisons using the IBM WebSphere Trade application (originally created by IBM as a best-practice performance application for IBM WebSphere) , Microsoft created an application that is functionally equivalent to the IBM WebSphere Trade application, both in terms of user functionality and middle-tier database access, transactional and messaging behavior. This application was created using best-practice programming techniques for .NET and the Microsoft Application Development platform. The resulting application, the ***.NET StockTrader***, is now published on MSDN as a best-practice .NET enterprise application at http://msdn.microsoft.com/stocktrader. The .NET StockTrader is a service-oriented application that

utilizes Windows Communication Foundation (WCF) for its underlying remoting and messaging architecture. The user interface is an ASP.NET/C# Web application that is equivalent to the IBM WebSphere Trade Java Server Pages (JSP) application. Additionally, the middle-tier services, written in C#, mirror the functionality and transactional characteristics of the backend IBM WebSphere Trade services which are based on JAVA EE 5 and the IBM WebSphere 7 application server. Microsoft also modified the IBM WebSphere Trade 6.1 application to take advantage of IBM WebSphere 7, including the use of a streamlined JSP/Servlet/JDBC architecture (vs. EJBs), and the latest JAX-WS Java Enterprise Web Services software stack.

For the benchmark comparisons using the WSTest workload, Microsoft created the equivalent Web Services using IBM WebSphere 7 and JAX-WS; as well as a C# implementation based on Microsoft .NET and WCF Web Services. These sources are also included in the .NET StockTrader download on MSDN.

Multiple Clients with Open Integration to Middle Tier via WCF

As a service-oriented application based on WCF, multiple interoperability scenarios are enabled with the .NET StockTrader. Since both the IBM WebSphere 7-based Trade application and the .NET StockTrader application expose their middle-tier services as industry-standard Web Services, the two applications can be seamlessly integrated with no code changes required. The JSP Trade front-end application can fully utilize the .NET middle tier services and messaging capabilities of the .NET StockTrader; and the ASP.NET StockTrader front-end application can fully utilize the Java-based WebSphere 7 Trade middle tier services and messaging capabilities. This interoperability is possible with the .NET StockTrader since WCF, Microsoft's component remoting and distributed application programming model, is fundamentally based on open Web Service standards including SOAP, XML and the latest WS-* industry standards. In addition to the ASP.NET Web based application that integrates via services with the middle tier, the sample also includes a Windows Presentation Foundation (WPF) desktop client (also developed in C#), that provides a smart-client interface to the middle tier. The WPF client can also seamlessly connect to either .NET middle tier services, or the IBM WebSphere 7 Trade middle tier services simply by changing the services URL in the configuration page—no code changes are required to achieve this interoperability.

Full Disclosure Notice

The complete source code, test script flow and all testing methodology for this benchmark are available online. Any reader may download and view the actual code for all implementations tested, and may further perform the benchmark for themselves to verify the results. The benchmark kit can be downloaded from http://msdn.microsoft.com/stocktrader. Extensive time was taken to generate results that represent optimal tuning for the platforms tested, and we are quite confident in the results. We encourage customers to download each kit and perform their own comparative testing and functional and technical reviews of each workload tested.

Middle-Tier Hardware Platforms Tested

A key goal of the testing was to document the performance and relative cost of IBM WebSphere 7 running on an IBM Power 570 server (Power6-based) vs. IBM WebSphere 7 on a Hewlett Packard BladeSystem (Intel-based) running Microsoft Windows Server 2008 R2; and then to compare these results to the identical workloads running on the HP BladeSystem with Microsoft .NET and Windows Server 2008 R2. Complete pricing details, including hardware and software costs, are documented in Appendix A of this paper. For the first two workloads, which include database access, each platform was tested against the same database hardware and disk configuration, which was carefully monitored to ensure the databases were never a bottleneck for any middle-tier platform tested. In addition, network monitoring was carefully conducted to ensure the network was never an artificial bottleneck. Hence, the performance measured accurately captures the peak TPS rate achieved on the middle tier hardware and software platform tested. **In all cases, server CPU saturation rates were at or above 96% CPU utilization at peak TPS, reflecting the fact no external bottleneck (database, network) artificially limited the results achieved.**

Additionally, in all testing, database CPU processor utilization remained below 50%, with no significant disk queuing.

IBM Power 570 (based on Power6 Architecture) running IBM AIX 5.3

- 8 IBM Power6 cores, operating at 4.2GHz
- 32 GB RAM
- AIX 5.3
- 4 x 1 GB NICs

Hewlett-Packard C7000 Blade System with 4 HP ProLiant BL460c blades

- 4 Hewlett Packard ProLiant BL460c blades
- Each blade has one Quad-Core Intel® Xeon® E5450 (3.00GHz, 1333MHz FSB, 80W) Processor
- Each blade has 32 GB RAM
- Each blade is running Microsoft Windows Server 2008 R2/64-bit
- Each blade has 2 x 1 GB NICs

For the Trade Web Application benchmark and the Trade Web Services benchmark, two separate databases are utilized to ensure enough database and disk I/O capacity to handle the large volume of transactions each platform is able to produce.

- For the IBM WebSphere 7 on IBM Power 570 platform tests for the Web application and Trade Web Services test, two instances of IBM WebSphere (nodes) are running on the IBM Power 570 server, each configured to communicate with a separate and dedicated backend trade database. IBM DB2 v9.5 Enterprise Edition (64-bit, Windows Server) is used for each of these databases, configured on identical hardware (see the database section for details).
- The WSTest benchmark does not utilize a database; a single instance of IBM WebSphere 7 was used on the Power 570 server and on each HP Blade.

- For the IBM WebSphere 7 on HP BladeSystem C7000 platform tests (Windows Server 2008 R2), one instance of IBM WebSphere is running per blade, with two blades communicating to one of the DB2 databases, and two blades communicating to the other dedicated IBM DB2 database.
- For the Microsoft .NET/Windows Server 2008 platform tests on the HP BladeSystem C7000, a single .NET 4.0 CLR instance is running on each blade. Two servers are communicating to one database, two servers to the second database. The databases used for the .NET tests are Microsoft SQL Server 2008, 64-bit, on Windows Server 2008 R2. They run on the exact same hardware (servers, disks) as the IBM DB2 9.5 databases used in the WebSphere 7 tests.

Test Methodology and Reporting Metrics

For each result, we report a peak sustained transaction per second (TPS) throughput rate as averaged over a 30 minute measurement period as tracked by the testing tool. For the Web Application performance test, HP Mercury LoadRunner was used to drive load and capture the results. For the two Web Service workloads (Trade Web Services and WSTest); the .NET Capacity Planner (provided for free with the .NET StockTrader download) was used to drive load and capture results.

Web Application Benchmark

In the Mercury LoadRunner test cases, 32 physical client machines (1.2 GHz, 512MB RAM) were used to run the test scripts, each client driving several hundred distinct users with a one-second think time.

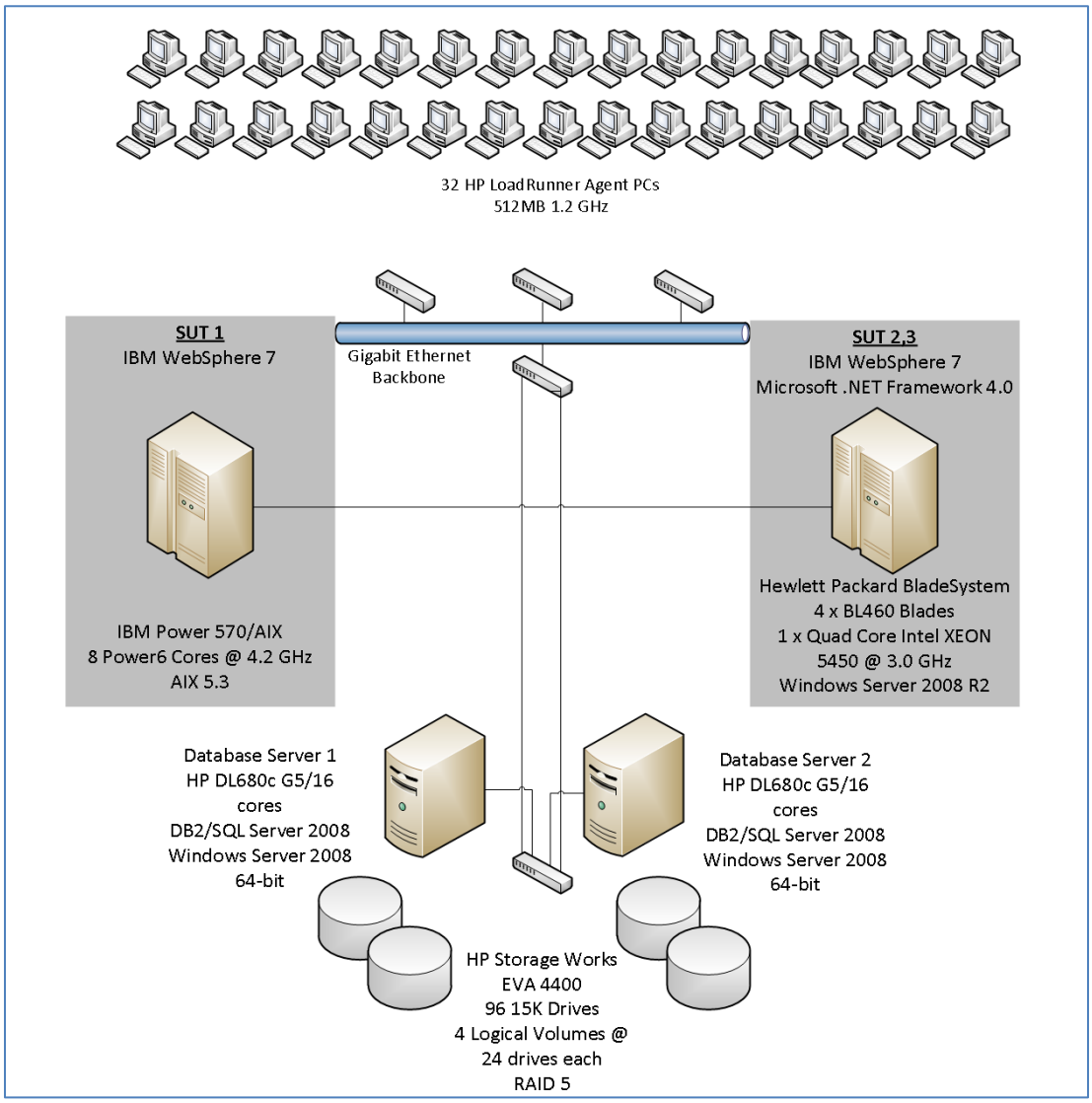


Figure 1: Test bed used for the Web Application benchmark using IBM WebSphere 7 Trade and Microsoft .NET StockTrader Web applications. There are two distinct hardware platforms that make up the System Under Test (SUT); however, the HP BladeSystem runs both the IBM WebSphere 7 Trade workload, and the Microsoft .NET StockTrader workload; representing three total SUTs. The databases are configured with plenty of capacity for all systems tested, and are not part of the SUTs.

Trade Middle Tier Web Service Benchmark and WSTest Web Service Benchmark

In the .NET Capacity Planner Web Service tests, 10 distinct clients were used to drive load (Intel quad-core systems @2.4 GHz) with a 0.1 second think time.

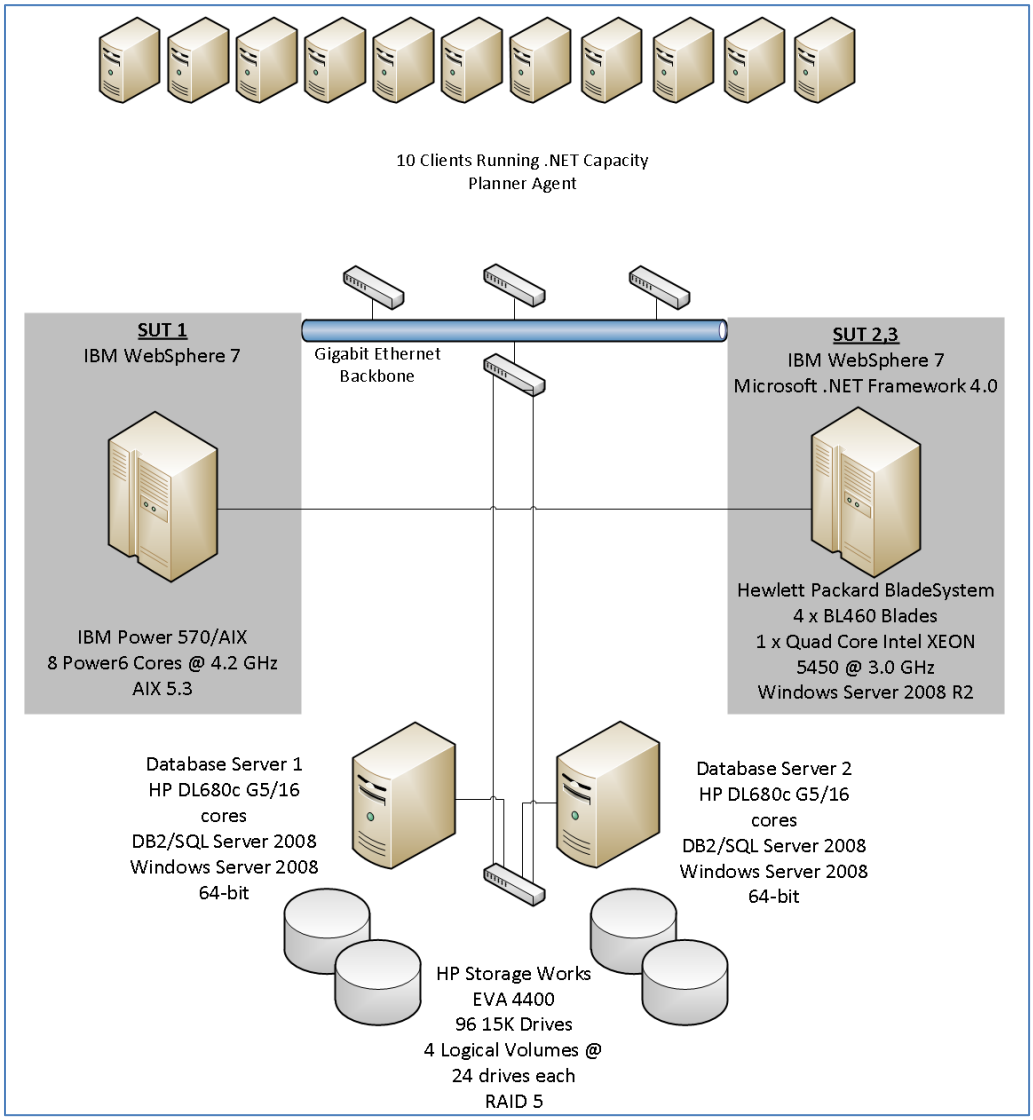


Figure 2: Test bed used for the two Web Service benchmarks. Note the databases are only utilized for the Trade Web Service test; as the WSTest benchmark involves no business logic or database access. There are two distinct hardware platforms that make up the System Under Test (SUT); however, the HP BladeSystem runs both the IBM WebSphere 7 Trade and Java WSTest workloads, and the Microsoft .NET StockTrader and .NET WSTest workloads; representing three total SUTs for each of the two Web Service tests. The databases are configured with plenty of capacity for all systems tested, and are not part of the SUTs.

Running the Benchmarks and Capturing Reported Metrics

Load balancing is achieved across the 4 HP blades (for both .NET and IBM WebSphere 7) using Windows Network Load Balancing, built into Microsoft Windows Server 2008. In all cases, users are added to the system with iterative test runs until peak throughput is obtained. Extensive benchmark runs were done

prior to measurement to ensure proper tuning of the middle tier systems. For each of the three workloads, and across all SUTs tested, we report the peak sustained ***TPS rate***, and also a ***calculated dollar cost per TPS*** (price performance metric) so customers can better understand what they are paying for that performance using a standard normalized metric: cost of platform divided by peak TPS rate measured. This calculation mirrors how testing organizations such as the Transaction Processing Council (TPC) report price performance metrics.

The cost calculations are based on the middle tier application server software licensing costs and hardware acquisition costs of the middle tier systems as tested. Notes and details on the pricing of the middle tier application servers are included in the Appendix, and prices are based on published pricing from each vendor or as purchased from authorized resellers.

Customers should understand that full .NET capabilities are included in every edition of Windows Server 2008 R2, and upgraded versions of .NET are made available for free download from MSDN (for example, .NET 4.0 as tested here). Hence, there is no additional or separate application server cost associated with a .NET application; while commercial JAVA EE 5 application servers such as WebSphere are separately licensed (and typically quite expensive) products. The dollar cost per TPS rates might, therefore, surprise some readers.

It is also important to remember that this is a test of specific workloads based on the IBM WebSphere Trade application and equivalent .NET StockTrader application; and the Java and .NET WSTest workloads. Each of these workloads, however, utilizes most (if not all) of the commonly deployed architectural building blocks used in common enterprise application scenarios.

.NET Framework 4.0 vs. .NET Framework 3.5

Finally, the .NET results presented here are for the current .NET Framework 4.0 version running on Windows Server 2008 R2 with IIS 7.5. For all tests, however, benchmark runs were also completed on .NET Framework 3.5, the previous version of the .NET Framework, also running on Windows Server 2008 R2. All .NET 4.0 results in these tests were equivalent in performance (with <5% difference in any test) to the .NET Framework 3.5 results.

Fair Benchmark Comparisons between .NET StockTrader and IBM WebSphere 7 Trade

Since each application supports many different configurations, it is very important to understand what constitutes a fair comparison. The configurations compared must be equivalent. This means that the applications must produce the exact same functionality and processing behavior in the configurations compared. You cannot, for example, compare one application running with caching and the other with no caching. The .NET StockTrader, while based on .NET and not JAVA EE 5, was designed to mirror most of the IBM Trade configurations possible with this testing goal in mind. The WebSphere 7-optimized Trade application, with Java source provided with the .NET StockTrader download, runs in the fastest configuration possible: Java Server Pages, Servlets, and straight JDBC database access with no Enterprise Java Beans.

Database Access Technology/Programming Model

WebSphere 7 Trade:

- JDBC

The JDBC mode eliminates the Entity Beans and CMP, and instead uses direct JDBC calls to the database. Java model classes are used to pass data between the logical tiers in the application.

.NET StockTrader:

- ADO.NET

The ADO.NET implementation uses C# model classes to pass data between tiers. It uses ADO.NET DataReaders isolated in a separate data access layer (DAL) as a best-practice performance programming practice, and to maintain clean separation of database logic from the other tiers of the application.

Caching

WebSphere 7 Trade

- WebSphere is configured for Servlet Caching, and only the Market Summary JSP fragment is cached. The cache is renewed on a 3 minute cycle (absolute expiration), as defined in the WebSphere Cachespec.xml file. All other elements of the application are not cached, database interactions occur on every request. This fragment caching is necessary, since the Market Summary SQL query is very heavyweight, and with any real data load, would quickly bring the database and the application to a crawl if run on every visit to the home page. Since the query is the same in the .NET StockTrader application, the same is true for .NET StockTrader. For benchmark runs, we tuned to cache to a 3 minute absolute expiration in both cases.

.NET StockTrader

The .NET StockTrader uses the .NET cache API (output caching specifically) to cache the Market Summary control (fragment) for 3 minutes (absolute expiration). We chose not to implement

further caching in the application because it is simply not realistic. While WebSphere Trade can cache stock quotes, account data, portfolio data and the like (and .NET StockTrader could too if implemented), the simple fact is that this is not a realistic approach or "cache policy" for this application. Consider that the IBM cache, while distributed (it can keep cached items in sync across clustered servers), is not invalidated by the data source itself. Hence, an update to a database table by any other application using the same database would result in possibly corrupt data, or at least presenting incorrect "stale" data to users in the application. Unless a customer is willing to direct ***all*** database updates/deletes/inserts through the same WebSphere instances, data on the middle tier would quickly become out of sync with the actual database. In other words, such a strategy keeps the organization from building new applications on other platforms against a common database. Market Summary information is fine to cache, since its read-only and can stand to be several seconds stale; user account balances and stock price information used on trades cannot be stale. Hence, the .NET StockTrader does not implement further caching beyond Market Summary.

Enable Long Run Support

IBM WebSphere 7 Trade

- Off

This setting was introduced to Trade with WebSphere Trade 6.1, apparently because customers running the benchmark for long periods of time saw steady degradation of performance as user accounts started to contain more and more orders. This resulted in heavier queries, and large and steadily increasing amounts of data being passed between tiers and formatted into the account summary page. Hence, IBM created this setting. However, since our test database is loaded with many more user accounts, neither the WebSphere 7 version of Trade or the .NET StockTrader have an issue in running benchmarks over extended periods of time under heavy load.

.NET StockTrader

There is no equivalent setting for .NET StockTrader; instead we ran the benchmark for both applications against a larger, much more realistic data load (500,000 accounts, 5 order per account, 100,000 quotes). We provide a database loader (written in .NET Windows Forms) to load both SQL Server 2008, Oracle 11G, and DB2 V9.5. This loader program will also reset the database between benchmark runs to the same starting state by deleting added data records.

Database Load

As discussed, we used a default load for all benchmark runs (reset between runs) of 500,000 accounts, 5 orders/holdings per account, and 100,000 quotes. This is much more realistic than the IBM default settings for Trade.

Database Configuration

The IBM WebSphere 7 Trade application was tested against an all-IBM configuration, using IBM DB2 V9.5 (Enterprise Edition) as the backend databases, and the latest IBM DB2 V9.5 JDBC drivers for data access. The .NET StockTrader was tested against an all-Microsoft setup, with backend SQL Server 2008 databases (Enterprise Edition). The benchmark is not a database benchmark: enough capacity was employed for the database hardware to ensure it was not a bottleneck in any benchmark run. For both the Trade Web Application performance benchmark and the Trade Web Services benchmark, two backend databases were deployed to eliminate possible database contention under high TPS rates. Each database was deployed to the same two 64-bit Windows Server 2008 servers. Storeage is a high-capacity, high-speed HP StorageWorks EVA 4400 Disk Array with a total of 96 15K drives in a RAID 5 configuration.

Database Server Specification

2 x HP BL680c G5 blades

- Each blade configured with 4 Quad-Core Intel XEON CPUs, @2.4GHZ (16 cores in each blade)
- Each blade configured with 64 GB RAM
- Each blade configured with 4 x 1GB NICs
- Each blade running IBM DB 9.5 Enterprise Edition 64-bit and Microsoft SQL Server 2008 64-bit
- Each blade running Microsoft Windows Server 2008 64-bit, Enterprise Edition
- Each blade has 2 4GB HBAs for fiber/sans access to the EVA 4400 storage.

Database Storeage Specification

- HP StorageWorks EVA 4400 Disk Array
- 96 15K drives total
- 4 logical volumes consisting of 24 drives each
- Database server 1: Logical Volume 1 for logging
- Database server 1: Logical Volume 2 for database
- Database server 2: Logical Volume 3 for logging
- Database server 2: Logical Volume 4 for database

All tuning steps were followed for DB2 according to the IBM Trade 6.1 documentation, however, additional logging space was configured given the larger data load. The equivalent drive space was configured for SQL Server logging. The database disk usage was closely monitored to ensure each run could complete without requiring the database to extend the logging or data file space during a benchmark run. This is an important consideration for custom tests.

Test Methodology for the Trade Web Application Benchmark

Test Scripts

For the Web Application test, Mercury LoadRunner was used to record test scripts—browser interactions that exercise most of the functionality in the application. These were run across 32 client

machines (500 MHz Windows XP desktops with 512 MB RAM). User agents were configured to run with a one second think time between each request. Each benchmark run included a warm up run to get to steady state, and a 30 minute measurement period. TPS rates were determined by LoadRunner by averaging across the 30 minutes. Error rates were monitored to ensure they remained at less than .01% during the measurement period. User loads were run for each application up to a number that represented peak throughput for that configuration, as determined in many iterative runs (literally hundreds) during the tuning stages. Extensive time was spent tuning IBM WebSphere (see the appendix) to achieve peak throughput for the software/hardware configuration tested. IBM does not publish pre-set tuning guides for Trade, and developers must iteratively test and tune the various knobs in WebSphere (there are many) to get to an optimal setup for a given hardware configuration and software workload. .NET does not require nearly as much tuning, and in general will scale quite well out of the box, given a properly coded application. Some tuning was applied, however, and this is documented in detail along with the WebSphere tuning in the appendix.

In general, it is fairly straightforward to recognize an in-properly tuned system for both platforms. Given enough database capacity and network capacity, each application server running under load should be able to reach close to full saturation (~100% CPU saturation) when properly tuned, assuming good vertical scaling of the technology/OS across processors, and no external bottlenecks. It is a requirement to iteratively test after adjusting, individually, the core tuning settings for the application, to determine peak throughput and user loads that achieve peak throughput (neither over saturating or under saturating the middle tier server being tested). With WebSphere 7, these tuning settings are extensive, including several different thread pools, connection pools for databases and queue connection factories, Java heap sizes, and the like. We are quite confident in the results and the system tuning applied, as in all tests servers under test achieved between 96% and 100% CPU saturation at peak TPS. However, should IBM recommend different (specific) settings, we will be happy to re-run the benchmark and publish new results with these settings. Customers can also run the benchmark—it is a great way to really judge the capacity of the two platforms for a realistic workload, and then to judge the cost of each platform and compare the cost to the results achieved.

Simulated User Settings

Mercury agents were set to not download images (this is not a web server/network I/O benchmark) during runs. They make requests to the application server, and all processing is completed and just the HTML returned to the agent. This reduces the overhead on the agent machines and network, and helps ensure the 32 client machines used in the testing never become an artificial bottleneck. Just as importantly, the agents were **configured to reset connections between iterations**, to simulate constantly new users logging into the application, and more fully exercise the underlying networking stacks and HTTP keep alive system that the application servers and Web servers use to support large concurrent user bases. Too many benchmarks are simply run with 10-15 threads, no think times and no network resets between script iterations. These types of benchmarks often produce very different (often over-inflated) results than real world usage conditions. Our settings are meant to much more closely mimic the real world. Think times (even if just one second) and connection resets between iterations make all the difference here.

IBM HTTP Server vs. Port 9080

When benchmarking IBM WebSphere, it is important to understand that while WebSphere provides an in-process HTTP listener service (port 9080, by default), IBM best practice recommended deployments are in conjunction with the full-blown IBM HTTP Server (a repackaged version of Apache). Hence, for all configurations, we used IBM HTTP Server as packaged with WebSphere, configured with the WebSphere Plug-in. On the .NET side, we used Microsoft Internet Information Services 7.0 (IIS 7), also co-located with the server being tested. IBM HTTP Server tuning details for Windows and AIX are included in the Appendix.

Web Application Pages Exercised by the Test Scripts

The test scripts were designed to drive load on the system in a way that exercises most functionality in the application, and puts a heavier emphasis on transactions; such as adding new registered users and buying stocks. The precise flow of the test scripts (exactly the same for all test runs on all platforms) is listed below. A one second think time was used between all URL requests, and in the results, a 'transaction' is defined as the successful completion of the URL request to the server, with valid response/HTML returned.

Part 1 (Existing User)

- Login random registered user (1 to 500,000 users loaded in database; the login includes in both apps a redirect to the home page, and all the logic to login and display home page)
- Request random quote (1 to 100,000 distinct quotes loaded in database; one post performed with 1 stock requested)
- Request random quote (1 to 100,000 distinct quotes loaded in database; one post performed with 1 stock requested)
- Visit Account Page (no account update performed)
- Request random quote (1 to 100,000 distinct quotes loaded in database; one post performed with 1 stock requested)
- Visit Portfolio Page
- Visit Home Page
- Request random quote (1 to 100,000 distinct quotes loaded in database; one post performed with 1 stock requested)
- Request random quote (1 to 100,000 distinct quotes loaded in database; one post performed with 1 stock requested)
- Logout the Registered user via logout page

Part 2 (New User—run in sequence after Part 1)

- Register a new user/submit registration form (this also logs new user in with redirect/display of home page)
- Request random quote (1 to 100,000 distinct quotes loaded in database; one post performed with 1 stock requested)
- Visit Account Page (no account update performed)

- Request random quote (1 to 100,000 distinct quotes loaded in database; one post performed with 1 stock requested)
- Buy a random stock symbol (1 to 100,000 stock symbols in database; buy operation involves a direct post/submit to the order submission pages, which submit the order for all backend processing)
- Visit Portfolio Page
- Request random quote (1 to 100,000 distinct quotes loaded in database; one post performed with 1 stock requested)
- Buy a random stock symbol (1 to 100,000 stock symbols in database; buy operation involves a direct post/submit to the order submission pages, which submit the order for all backend processing)
- Request random quote (1 to 100,000 distinct quotes loaded in database; one post performed with 1 stock requested)
- Buy a random stock symbol (1 to 100,000 stock symbols in database; buy operation involves a direct post/submit to the order submission pages, which submit the order for all backend processing)
- Visit Account Page (no account update performed)
- Request random quote (1 to 100,000 distinct quotes loaded in database; one post performed with 1 stock requested)
- Buy a random stock symbol (1 to 100,000 stock symbols in database; buy operation involves a direct post/submit to the order submission pages, which submit the order for all backend processing)
- Visit Portfolio Page
- Visit Home Page
- Logout

Test Methodology for the Trade Web Services Benchmark and WSTest Web Service Benchmark

Test Scripts

For these tests, the .NET Capacity Planner tool (a Web Service load-generating client included with the .NET StockTrader download) was used to capture average TPS rates over a 30 minute measurement interval. 10 quad-core systems were used to drive load, each running the agent software. Each agent was running between 50 and 300 user threads, operating at a 0.1 second think time. Iterative testing was done to determine user loads that resulted in near 100% server saturation and peak sustained TPS rates over the 30 minute measurement interval. For the Trade Web Services tests, the Capacity Planner was set to use the Transaction-Light script, of the three selectable scripts in the tool. For the WSTest tests, the message size was set to 20.

IBM HTTP Server vs. Port 9080

As with the Trade Web Application benchmark, IBM WebSphere was configured with the IBM HTTP Server. Similarly, the .NET WCF services were hosted via the IIS 7 web server.

Benchmark Results

Trade Web Application Test

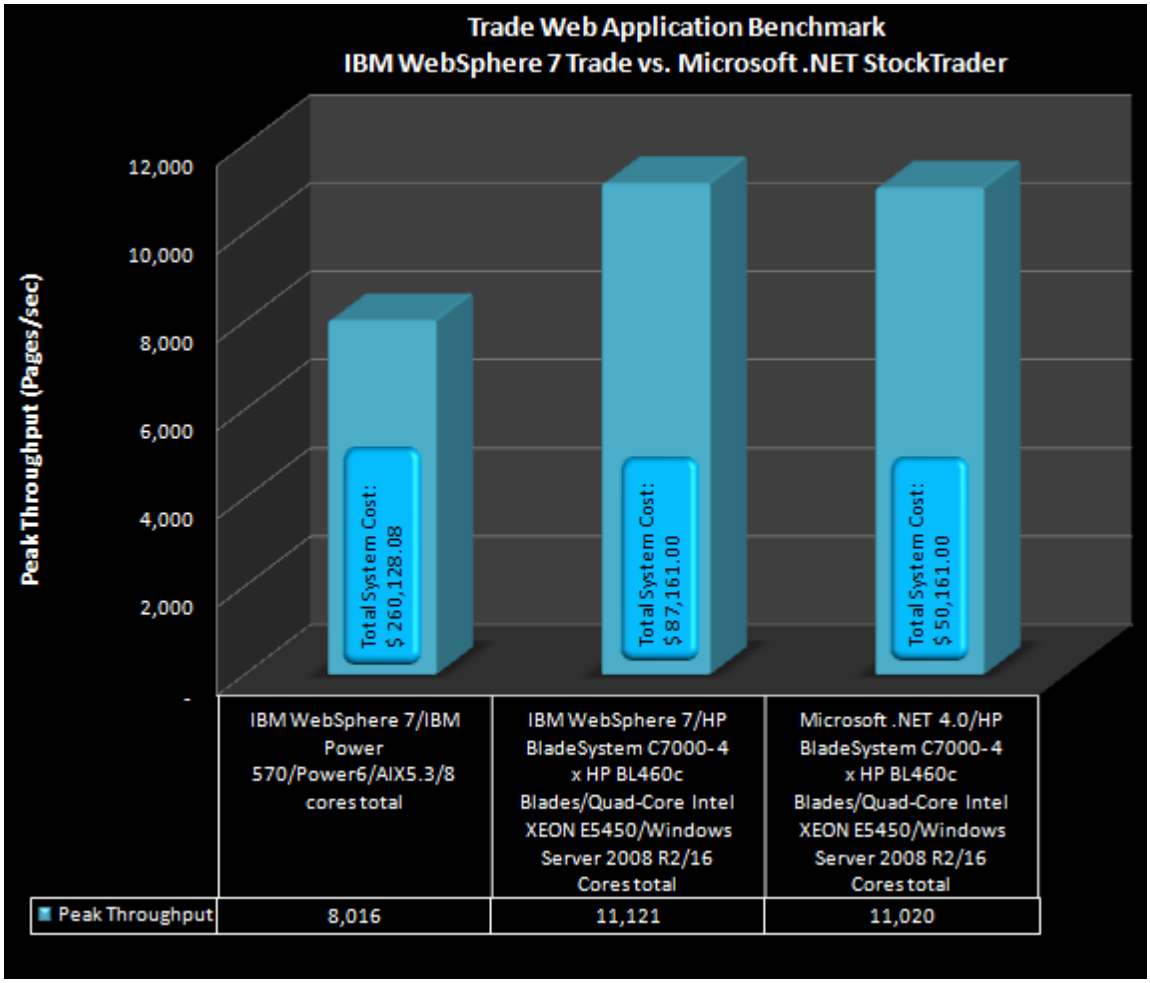


Figure 3: Peak TPS Rates for the Web Application Test. IBM WebSphere 7 Trade is running using JSPs/Servlets with direct JDBC access to the databases. Microsoft .NET StockTrader is running using ASP.NET/Web Forms and ADO.NET access to the databases. Mercury LoadRunner is used to drive Http requests to the system under test, with the html pages returned to the virtual users.

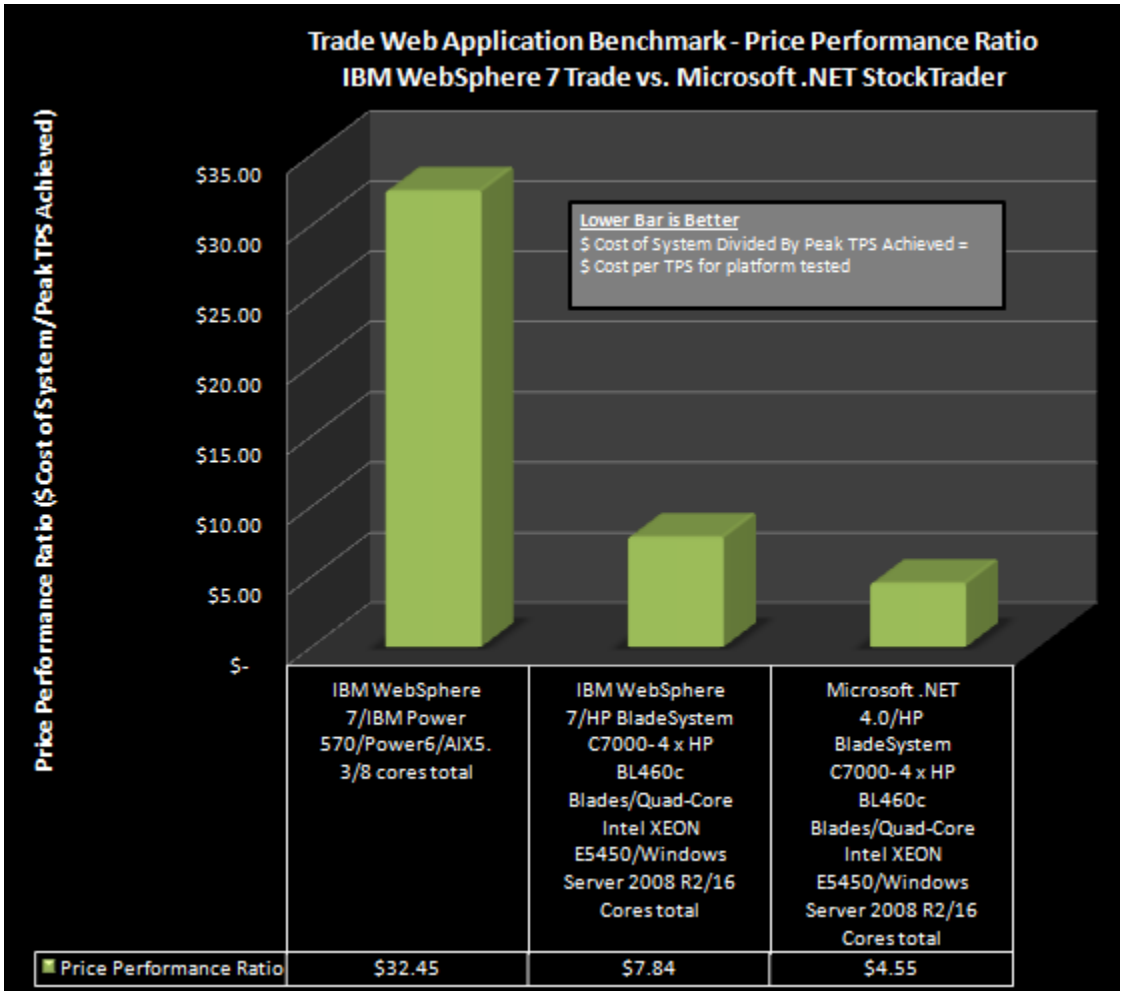


Figure 4: Price Performance Ratio for the Web Application Test. See appendix for pricing calculations, which includes the hardware and software costs of the middle tier application server as tested.

Trade Web Application Benchmark Discussion

This benchmark tests the complete Web applications for the IBM WebSphere 7 version of the Trade benchmark; and the .NET StockTrader benchmark. As such it shows the relative performance of a complete data-driven Web application across the three platforms tested. For both applications, the Web tier directs business service operations to the business service classes, and for most pages, there are several such operations per page, each with distinct database calls. In both applications, calls into the business service class from the Web tier and from the business service class into the data access class are made synchronously, and there are the same number of calls for each equivalent page across .NET and Java applications.

For the IBM WebSphere 7 version, a combination of Java Server Pages and Servlets are used to create the data-driven HTML pages for the user. The primary Servlet invokes a logical business service class which in turn invokes a separate database access class. The database access class in turn uses IBM's JDBC driver for DB2 v9.5 to directly access the remote IBM DB2 databases. The entire application is executing within an IBM WebSphere JVM instance per logical WebSphere node.

For the .NET StockTrader, ASP.NET Web Forms are used to invoke a logically partitioned business service class, which in turn accesses a logically partitioned data access class. ADO.NET is used with the .NET Provider for SQL Server to access the Microsoft SQL Server 2008 databases. The entire application is executing within a single .NET CLR worker process instance per .NET/Windows Server 2008 node.

The following conclusions can be drawn for this benchmark workload:

- .NET on Windows Server 2008 running on the Hewlett Packard BladeSystem C7000 outperforms IBM WebSphere 7 on the IBM Power 570 server by **37%**, yet the IBM Power 570 server costs **419%** more (in absolute dollars) than the HP BladeSystem running Microsoft .NET and Windows Server 2008.
- IBM WebSphere 7 on Windows Server 2008 running on the Hewlett Packard BladeSystem C7000 outperforms IBM WebSphere 7 on the IBM Power 570 server by **39%**, yet the IBM Power 570 server costs **198%** more (in absolute dollars) than the HP BladeSystem running Microsoft .NET and Windows Server 2008.
- Overall .NET on Windows Server 2008 shows by far the best price performance metric. IBM WebSphere 7 on IBM Power 570 has a price performance metric over 8 times higher than Microsoft .NET on Windows Server 2008. This means that for the performance actually delivered, IBM WebSphere 7 on the IBM Power 570 server **is actually over 7 times as expensive** (in dollars per TPS) than running the same workload on Microsoft .NET and Windows Server 2008 deployed to the Hewlett Packard BladeSystem.

Trade Middle Tier Web Services Test

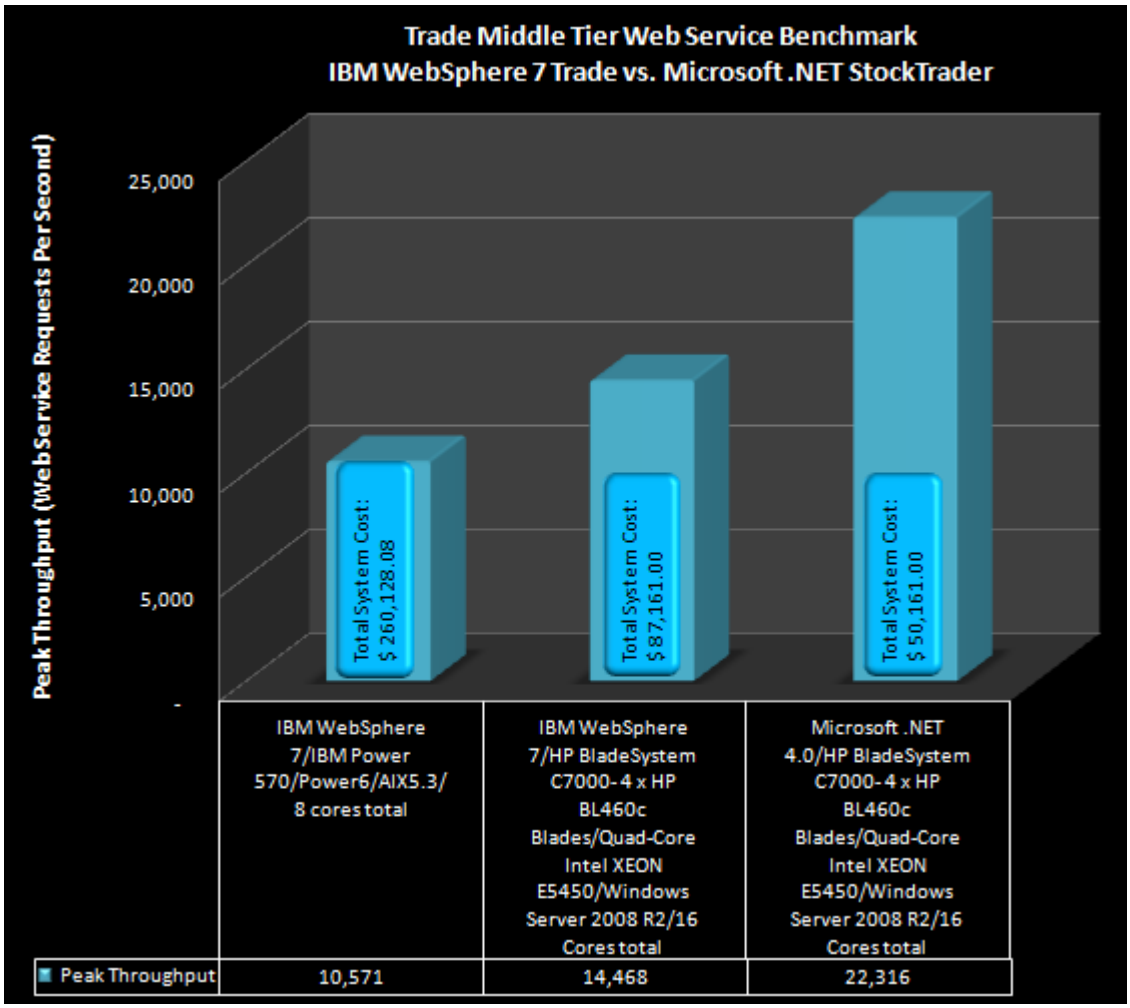


Figure 5: Peak TPS Rates for the Trade Middle Tier Web Service Test. The .NET Capacity Planner Web Service test tool is used to drive Web Service requests (SOAP requests) to the server, with the web service responses returned to the virtual users.

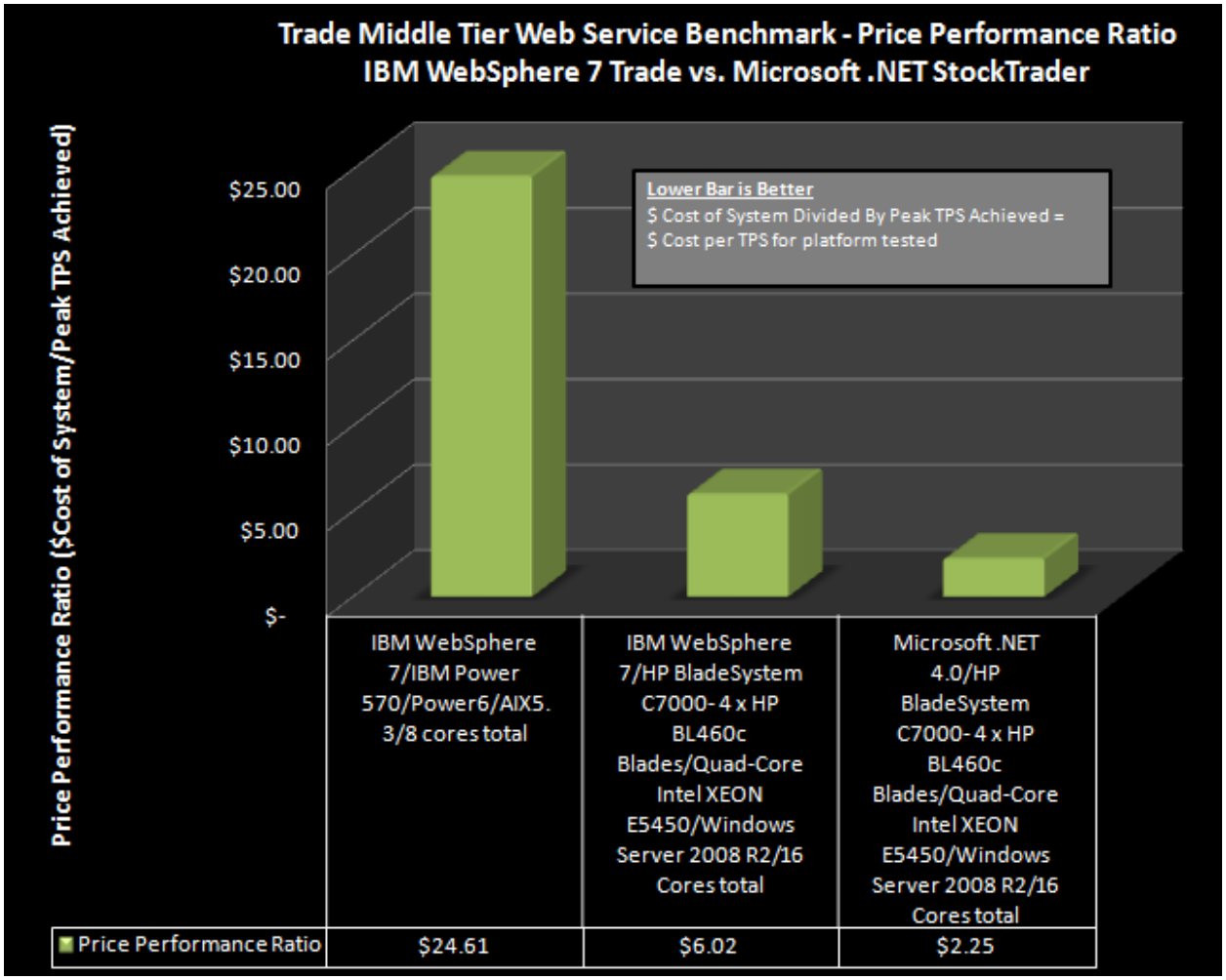


Figure 6: Price Performance Ratio for the Web Service Test. See appendix for pricing calculations, which includes the hardware and software costs of the middle tier application server as tested.

Trade Middle Tier Web Services Benchmark Discussion

This test is a Web Service workload, with backing business logic and database access and database transactions defined within the middle tier Web Service operations. This benchmark also tests the Trade/.NET StockTrader application; however, it tests the middle tier service operations via remote Web Service access. The service tier consists of a Web Service façade built on top of the business service operations of the IBM Trade and .NET StockTrader applications. It is important to note that in this benchmark, unlike the previous Web Application benchmark, the definition of a "transaction" (for TPS rates) is finer-grained than in the previous test. In this test, the peak throughput is measured on all platforms tested as a **single remote** Web Service request against the middle tier business operations. In the previous test, the throughput is measured as complete web page being returned to the benchmark clients; and each Web page has on average **multiple in-process** requests to the middle tier, with no XML/Web Service serialization; but on average more business service operations and database queries per "transaction" (web page returned). Hence, the TPS rates of the two different benchmarks should not be directly compared. For both the Java and .NET applications, for instance, we observe higher overall throughput rates in this test than the previous test; but that is due to the fact multiple business

service operations are being called in the Web Application Test (in-process) for each page returned; vs. this test measuring just a single business service tier request per Web Service (remote) call.

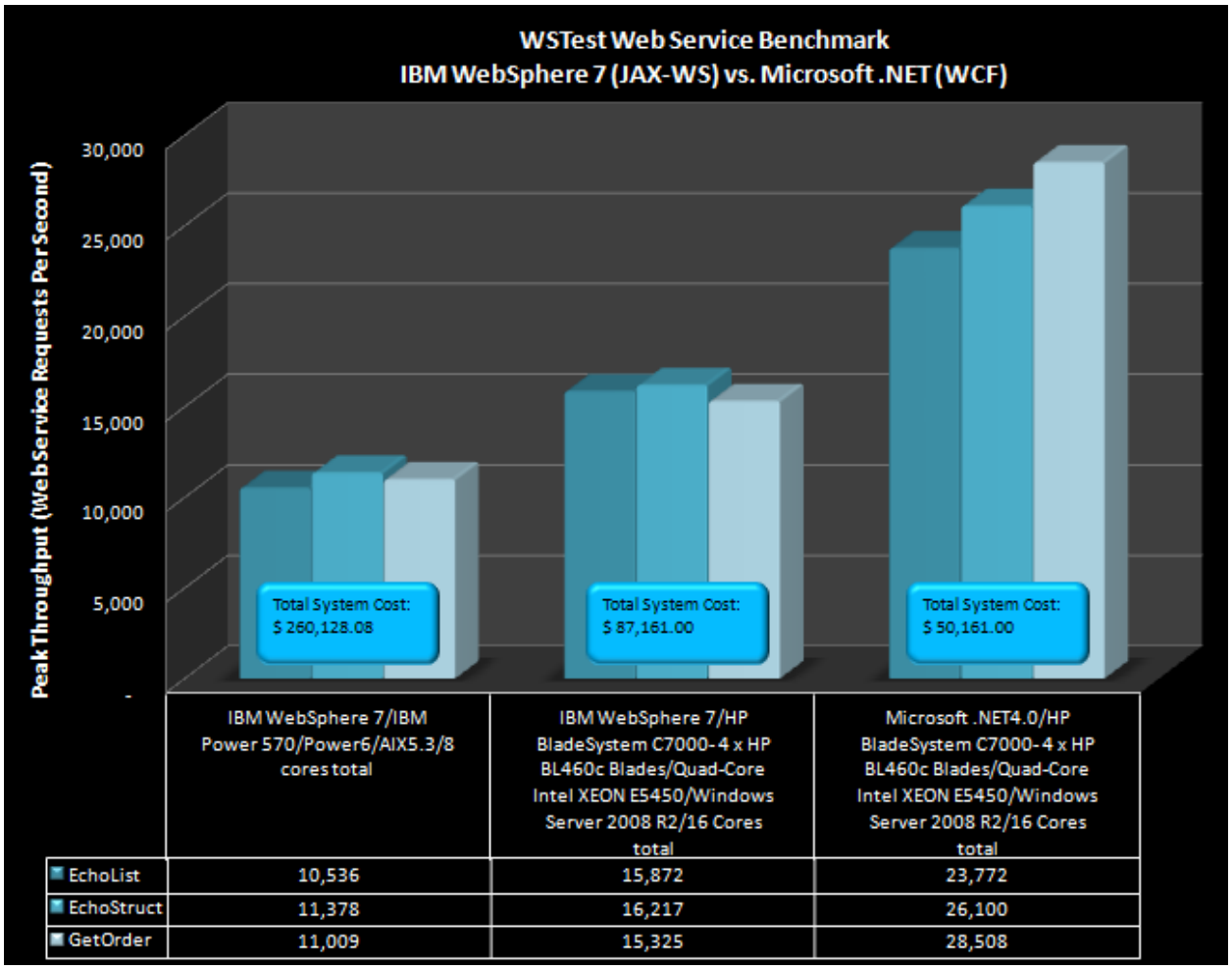
For IBM WebSphere 7, the Web Service façade is based on JAX-WS, and exposes the services over industry standard SOAP/WSDL. IBM HTTP Server is used to direct requests to the Web Service tier running with the IBM WebSphere 7 application server. For the .NET StockTrader, the Web Service façade is based on Windows Communication Foundation (WCF); which also exposes the services over the same industry standards for SOAP/WSDL. Internet Information Services 7.0 (IIS) is used to direct requests to the WCF-exposed services. Each application uses the exact same service and data contracts; and hence a single Web Service client can seamlessly access either tier with no changes required.

In this test, the .NET Capacity Planner Web Service test tool (included with the .NET StockTrader download, including source code) is used to drive load against the service tiers. 10 quad-core agents are configured to drive as much load as required to achieve peak throughput for the service tier being tested (without over saturating the servers). The Web Service responses (in the form of SOAP messages) are returned to the clients. The script flow is defined as the Transaction-Light benchmark within the .NET Capacity Planner, which tests the range of service operations, with less emphasis placed on buys than on other service operations (to reduce database load; which helps prevent DB2 or SQL Server disk I/O from becoming an external bottleneck).

The following conclusions can be drawn for this benchmark workload:

- .NET on Windows Server 2008 running on the Hewlett Packard BladeSystem C7000 outperforms IBM WebSphere 7 on the IBM Power 570 server by **111%**, yet the IBM Power 570 server costs **419%** more (in absolute dollars) than the HP BladeSystem running Microsoft .NET and Windows Server 2008.
- IBM WebSphere 7 on Windows Server 2008 running on the Hewlett Packard BladeSystem C7000 outperforms IBM WebSphere 7 on the IBM Power 570 server by **37%**, yet the IBM Power 570 server costs **198%** more (in absolute dollars) than the HP BladeSystem running Microsoft .NET and Windows Server 2008.
- Overall .NET on Windows Server 2008 shows by far the best price performance metric. IBM WebSphere 7 on IBM Power 570 has a price performance metric over 10 times higher than Microsoft .NET on Windows Server 2008. This means that for the performance actually delivered, IBM WebSphere 7 on the IBM Power 570 server **is actually over 10 times as expensive** (in dollars per TPS) than running the same workload on Microsoft .NET and Windows Server 2008 deployed to the Hewlett Packard BladeSystem.

WSTest Benchmark



	IBM WebSphere 7/IBM Power 570/Power6/AIX5.3/8 cores total	IBM WebSphere 7/HP BladeSystem C7000-4 x HP BL460c Blades/Quad-Core Intel XEON E5450/Windows Server 2008 R2/16 Cores total	Microsoft .NET4.0/HP BladeSystem C7000-4 x HP BL460c Blades/Quad-Core Intel XEON E5450/Windows Server 2008 R2/16 Cores total
EchoList	10,536	15,872	23,772
EchoStruct	11,378	16,217	26,100
GetOrder	11,009	15,325	28,508

Figure 7: Benchmark TPS rates for the WSTest benchmark, including EchoList, EchoStruct and GetOrder service operations. This benchmark, which involves no business logic or database access, isolates the performance of the Web Service stack within the application server (XML serialization, http network access, etc). The .NET Capacity Planner Web Service test tool is used to drive Web Service requests (SOAP requests) to the server, with the web service responses returned to the virtual users.

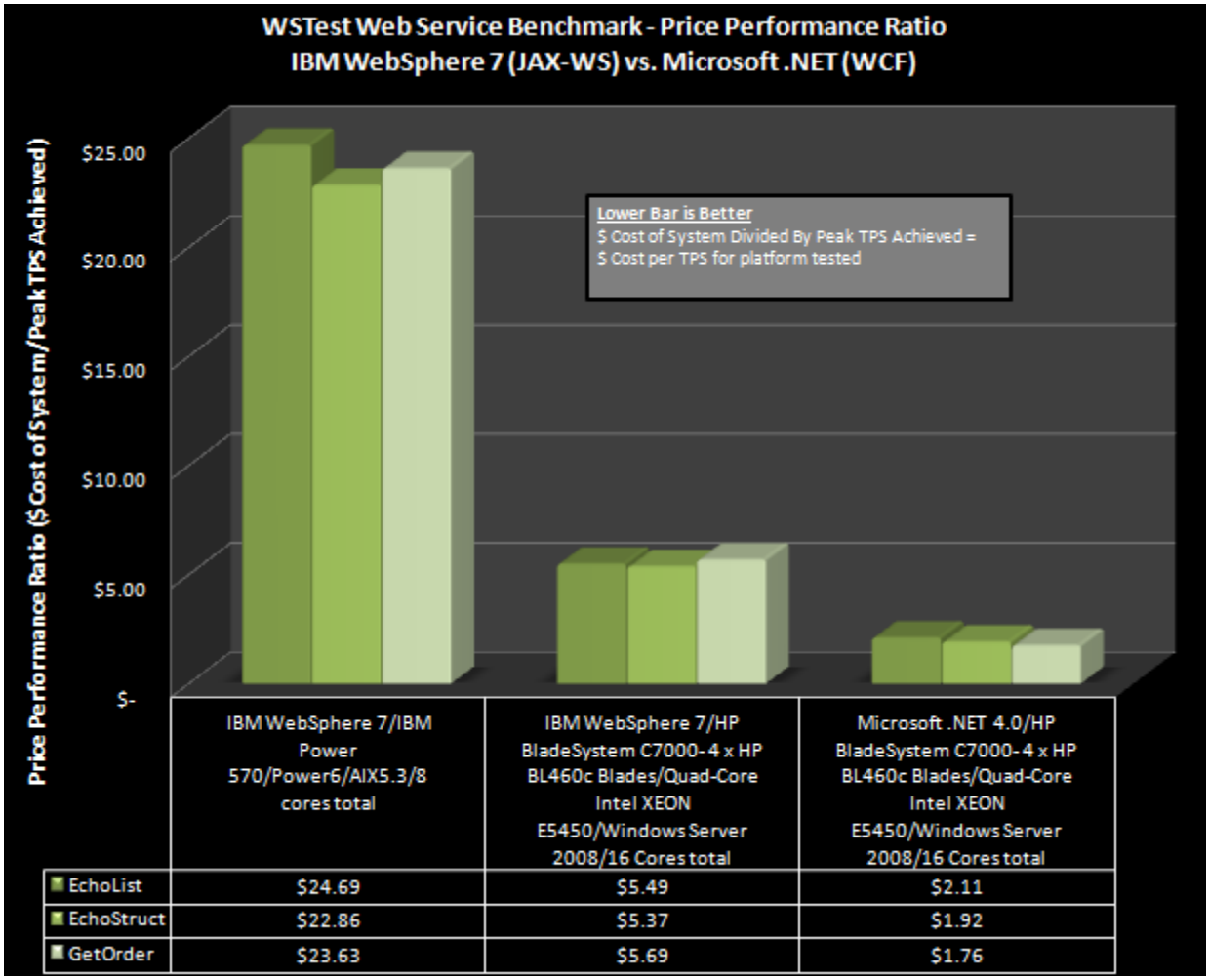


Figure 8: Price performance ratios for the WSTest benchmark. See appendix for pricing calculations/detail, which includes the hardware and software costs of the middle tier application server as tested.

WSTest Benchmark Discussion

This test is a Web Service workload, with **no** backing business logic or database access defined within the middle tier Web Service operations (as opposed to the previous test, which has both). This benchmark is based on the WSTest workload, a Web Service benchmark originally designed by Sun Microsystems and augmented by Microsoft. The service tier consists of several different operations (EchoList, EchoStruct and GetOrder). With no business logic, these operations isolate just the performance of the Web Service software within a given platform, including network access, and XML serialization/deserialization of SOAP messages and objects. In this test, across all platforms, we notice that TPS rates produced for each platform are roughly equivalent to those produced in the previous Trade middle-tier Web service test. Readers might ask why, if there are no business logic operations or database queries in the WSTest benchmark, are roughly equivalent TPS rates achieved in this test when compared to the previous test?

The answer is that the Web Service message payload in this test, on average, is larger than in the previous Trade middle tier Web Service test. Hence, in this test while less work is being performed by

business logic and database access (there is none); more work is being done on XML serialization/de-serialization due to the larger message size. Hence, readers should not directly compare results of this test to the previous Trade middle tier Web Service test. While both tests involve remote Web Service operations, they are independent tests with different workload characteristics. However, it should be noted the WSTest benchmark in this test used a default message size of 20 (20 repeating objects in all returned message payloads); and readers are free to use the .NET Capacity Planner tool to test different message sizes if they desire. They will then observe all platforms (.NET, WebSphere) producing higher TPS rates with smaller message sizes, and proportionally lower TPS rates with larger message sizes, due to the relative increase or decrease in the message size and hence XML serialization costs.

For IBM WebSphere 7 implementation of WSTest, the Web Services are based on JAX-WS, and exposed over industry standard SOAP/WSDL. IBM HTTP Server is used to direct requests to the Web Service tier running with the IBM WebSphere 7 application server. For the .NET implementation of WSTest, the Web Services are based on Windows Communication Foundation (WCF); which also exposes the services over the same industry standards for SOAP/WSDL. Internet Information Services 7.0 (IIS) is used to direct requests to the WCF-exposed services. Each application uses the exact same service and data contracts; and hence a single Web Service client can seamlessly access either tier with no changes required.

In this test, the .NET Capacity Planner Web Service test tool (included with the .NET StockTrader download, including source code) is used to drive load against the service tiers. 10 quad-core agents are configured to drive as much load as required to achieve peak throughput for the service tier being tested (without over saturating the servers). The Web Service responses (in the form of SOAP messages) are returned to the clients. The script flow uses a message size of 20 (meaning 20 repeating elements within the SOAP message in each test).

The following conclusions can be drawn for this benchmark workload:

- .NET on Windows Server 2008 running on the Hewlett Packard BladeSystem C7000 outperforms IBM WebSphere 7 on the IBM Power 570 server by roughly **120%** (depending on the specific WSTest operation compared), yet the IBM Power 570 server costs **419%** more than the HP BladeSystem running Microsoft .NET and Windows Server 2008.
- IBM WebSphere 7 on Windows Server 2008 running on the Hewlett Packard BladeSystem C7000 outperforms IBM WebSphere 7 on the IBM Power 570 server by **51%**, yet the IBM Power 570 server costs **198%** more than the HP BladeSystem running Microsoft .NET and Windows Server 2008.
- Overall .NET on Windows Server 2008 shows by far the best price performance metric. IBM WebSphere 7 on IBM Power 570 has a price performance metric over 10 times higher than Microsoft .NET on Windows Server 2008. This means that for the performance actually delivered, IBM WebSphere 7 on the IBM Power 570 server **is actually over 10 times as expensive** (in dollars per TPS) than running the same workload on Microsoft .NET and Windows Server 2008 deployed to the Hewlett Packard BladeSystem.

Conclusion

This paper presents an extensive array of benchmark comparisons between IBM WebSphere 7 and .NET 4.0/Windows Server 2008 running three different application server workloads. The results show that running IBM WebSphere 7 on an Hewlett Packard BladeSystem C7000 with Windows Server 2008 offers significantly better throughput than running the equivalent workloads on an IBM Power 570 server with AIX---and at roughly 1/3rd the cost of the middle tier hardware and software. In addition, the tests show that the best throughput and lowest cost overall is Microsoft .NET running on Windows Server 2008—offering up to 111% better performance than the IBM Power 570; and at roughly 1/5th the total cost of the middle tier hardware and software. With published source code for all benchmark workload implementations and a free test tool provided, we encourage customers to perform their own comparative testing on the platform of their choosing; and also to use the .NET StockTrader application as a learning sample for various features of .NET 4.0, ASP.NET and WCF Microsoft enterprise development technologies.

Appendix A: Pricing

The following pricing was used as the base pricing and price performance metric calculations. Pricing is based on published suggested retail pricing obtained directly from the vendor, or quotes obtained by authorized resellers. Pricing does not include sales tax.

Hardware and Operating System Pricing

IBM Power 570

Item	Quantity	Price	Extended Price
9117-MMA IBM SYSTEM P 570	1	$6,932.60	$6,932.60
9117-MMA-0265 AIX PARTITION SPECIFY	1	NA	NA
9117-MMA-0266 LINUX PARTITION SPECIFY	14	NA	NA
9117-MMA-1845OPERATOR PANEL	1	$135.32	$135.32
9117-MMA-3646 73 GB 15K RPM SAS DISK DRIVE	4	$448.12	$1,792.48
9117-MMA-3660 PROC. FABRIC CABLE,2	1	$1,360.00	$1,360.00
9117-MMA-4650 RACK INTEG.-NOT FACTORY INTEG	1	NA	NA
9117-MMA-5622 4.2GHZ POW6-2CORE 12 DDR2	4	$5,202.00	$20,808.00
9117-MMA-5625 PROC. POWER REGULATOR	6	$1,020.00	$6,120.00
9117-MMA-5626 SYSTEMCEC ENCLOSURE WITH BEZEL	2	$340.00	$680.00
9117-MMA-5628 AC POWER SUPPLY, 1600 W	4	$1,021.36	$4,085.44
9117-MMA-5629 MEDIA ENCLOSURE AND BACKPLA	1	$125.80	$125.80
9117-MMA-5636 2X-1GB VIRTETH-INTEGR	2	$271.32	$542.64
9117-MMA-5648 SERVICE INTERFACE CARD	2	$680.00	$1,360.00
9117-MMA-5657 SERVICE INTERFACE CABLE	1	$1,360.00	$1,360.00
9117-MMA-5663 PROC ENCLOSURE AND BACKPLAN	2	$1,360.00	$2,720.00
9117-MMA-5666 I/O BACKPLANE	2	$3,060.00	$6,120.00
9117-MMA-5667 SYSTEM MIDPLANE	2	$680.00	$1,360.00
9117-MMA-5668 SAS DISK BACKPLANE-26SLOT	2	$714.68	$1,429.36
9117-MMA-5672 1 PROC ACTIV FOR PROC FT	8	$10,404.00	$83,232.00
9117-MMA-5680 ACTIV OF 1GB DDR2 POWER 6 MEM	32	$1,030.20	$32,966.40
9117-MMA-5693 0/4GB(4X1GB),667MHZ,DDR2,POW6	8	$1,032.24	$8,257.92
9117-MMA-5699 SYSTEM SHIP GROUP	1	$13.60	$13.60
9117-MMA-5736 PCI-X DDR 2.0	2	$528.36	$1,056.72

DUAL CHAN.U320			
9117-MMA-5767 2-PORT 10/100/1000-TX EXP ADP	4	$475.32	$1,901.28
9117-MMA-6671 PWR CORD(9FT),DRW.TO IBM PDU	4	$12.92	$51.68
9117-MMA-7164 IBM/OEM RACK-MOUNT DRW R KIT	2	$150.96	$301.92
9117-MMA-7870 POWER DISTRIBUTION BACKPLANE	2	$180.20	$360.40
9117-MMA-7942 ADVANCED PWR VIRTUALIZATION	8	$890.80	$7,126.40
9117-MMA-9300 LANGUAGE GROUP SPCF-US ENG	1	NA	NA
7212-103 STORAGE DEVICE ENCLOSURE	1	$867.65	$867.65
7212-103-1103 DVD-RAM 2 DRIVE	2	$536.00	$1,072.00
7212-103-1215 RACK MOUNT KIT	1	$100.50	$100.50
7212-103-9006 PSERIES INSTALL	1	NA	NA
7212-103-9300 ENGLISH PUBLICATIONS	1	NA	NA
7212-103-9764 4.5 M VHDCI/HD68 SCSI CABLE	2	NA	NA
7212-103-9860 RACK POWER CORD, AC, SINGLE	1	NA	NA
5692-A5L PSERIES SPO	1	NA	NA
5692-A5L-0999 5765-G34 VIRTUAL I/O SERVER	1	NA	NA
5692-A5L-0999 5765-G34 VIRTUAL I/O SERVER	1	NA	NA
5692-A5L-1101 BASIC N/C DVD MEDIA FEATURE	1	NA	NA
5692-A5L-2924 ENGLISH	1	NA	NA
5692-A5L-3435 OTHER MEDIA (AIX) DVD SYS W	1	NA	NA
5692-A5L PSERIES SPO	1	NA	NA
5692-A5L-0967 5765-G03 AIX 5L FOR POWER V5	1	NA	NA
5692-A5L-0968 5765-G03 EXPANSION PACK	1	NA	NA
5692-A5L-0970 5765-G03 AIX 5L UPDATE CD	1	NA	NA
5692-A5L-0975 MICROCODE UPD DISC TOOL V1.1	1	NA	NA
5692-A5L-0999 5765-G34 VIRTUAL I/O SERVER	1	NA	NA
5692-A5L-1100 DVD MEDIA PROCESS CHARGE	1	$50.00	$50.00
5692-A5L-2924 ENGLISH	1	NA	NA
5692-A5L-3435 OTHER MEDIA (AIX) DVD SYS W	1	NA	NA
5765-G03 AIX 5L FOR POWER V5.3	1	NA	NA
5765-G03-0005 AIX 5L FOR POWER V5.3 F5	8	$1,041.25	$8,330.00
5765-G34 VIOS 1.5	1	NA	NA
5765-G34-0002 VIRTUAL I/O SERV F5 W/HW	8	NA	NA
5771-SWM 1 YEAR SWMA PID	1	NA	NA

NUMBER			
5771-SWM-0496 SWMA FOR AIX 1 F5 SWMA 1Y RGST	8	$616.25	$4,930.00
5771-SWM-0498 SWMA FOR AIX 1 F5 SWMA 1Y7X24	8	$155.55	$1,244.40
5771-VIO VIRTUAL I/O SERVER SWMA 1YR	1	NA	NA
5771-VIO-0617 GRP F5 BASIC OTC	8	$93.50	$748.00
5771-VIO-0618 VIRTUAL I/O SERV F5 SWMA1Y724	8	$24.65	$197.20
7042-CR4 IBM 7042-CR4	1	$3,192.00	$3,192.00
7042-CR4-0962 HMC LMC V7	1	NA	NA
7042-CR4-4650 RACK INDICATOR-NOT FACT INTEGR	10	NA	NA
7042-CR4-4767 HMC CR4 REDUN POW SUPPLY 670	1	$184.80	$184.80
7042-CR4-6458 POW CBL, DRAW 14-FT, 250V/10A	2	$15.12	$30.24
7042-CR4-7802 ETH CBL,15M,HMC CSL TO SYSUNIT	1	$26.88	$26.88
7042-CR4-9069 HMC/SRV ORDER LINKAGE IND	1	NA	NA
7042-CR4-9300 LANG GROUP SPECIFY - US ENG	1	NA	NA
7316-TF3 FLAT PANEL CONSOLE KIT	1	$2,160.00	$2,160.00
7316-TF3-4650 NO FACTORY INTEGRATION IND	1	NA	NA
7316-TF3-8880 USB TRAVEL KEYB.W/CBL,US ENG	1	$108.00	$108.00
7316-TF3-9300 LANG GROUP SPEC-US ENGLISH	1	NA	NA
7316-TF3-9911 POWER CORD (4M) – ALL	1	NA	NA
5771-RS1 MCRSA FOR HMC 1 YEAR	1	NA	NA
5771-RS1-0612 MCRSA FOR HMC 1Y PRC MCRS 1Y	1	$212.50	$212.50
5771-RS1-0614 MCRSA FOR HMC PRC MCRS 1Y 7X24	1	$73.95	$73.95
5771-RS1-7000 Agreement for MCRSA	1	NA	NA
Total			**$215,728.08**

Hewlett Packard BladeSystem C7000

Hardware Pricing

HP Hardware pricing obtained from retail pricing via HP Direct, online at http://www.hp.com.

Windows Server 2008 pricing obtained based on Microsoft Suggested Retail Pricing at http://www.microsoft.com/windowsserver2008/en/us/pricing.aspx.

Item	Quantity	Price	Extended Price
Configurable- HP BladeSystem c-Class c7000 Enclosure 3-inch LCD - Three Phase	1	$13,821.00	$13,821.00
Details of included components with BladeSystem C7000 Enclosure			
HP BladeSystem c-Class c7000 Enclosure			
HP 3 Phase Power Module			
HP BLc7000 Management Module			
4 HP BladeSystem c7000 Fans			
8 ICE-BL 30 Day Evaluation Licenses - Insight Control Environment for BladeSystem			
HP c-Class Insight Control Environment for BladeSystem 8 License with 1 year of 24x7 Support			
2 GbE2c Ethernet Blade Switches for HP c-Class BladeSystem			
HP Redundant Onboard Administrator Option			
6 HP 2450W High Efficiency Power Supplies			
6 HP Active Cool Fan Option Kits			
Warranty - 3 years - parts, labor, onsite - next business day			
HP Care Pack 3-year, 4-Hour, 13x5 c7000 Enclosure Hardware Support	1	$752.00	$752.00
Configurable- HP ProLiant BL460c G5 Server Blade	4	$5,403.00	21,612.00
Details of included components with each HP BL460c Blade			
HP ProLiant BL460c G5 Server Blade			
Quad-Core Intel® Xeon® E5450 (3.00GHz, 1333MHz FSB, 80W) Processor			
HP 32GB Fully Buffered DIMM PC2-5300 8X4GB DR LP Memory			
Embedded Smart Array E200i SAS Array Controller (without cache)			
HP 64MB cache upgrade for E200i			
HP 72GB Hot Plug 2.5 SAS 10,000 rpm Hard Drive			
HP 72GB Hot Plug 2.5 SAS 10,000 rpm Hard Drive			
RAID 0 drive set (requires minimum 1 hard drive)			
2 Embedded NC373i Multifunction Gigabit Network Adapters			
HP iLO Advanced Blade 1 Server License with 1yr 24x7 Technical Support and Updates			
Integrated Lights-Out 2 (iLO2) Standard Blade Edition (integrated on motherboard)			
3-Year Limited Warranty Included			
HP Care Pack 3-year, 4-hour, 13x5 BL4xxc	4	$296.00	$1,184.00

Server Hardware Support; electronic:			
Subtotal Hardware			**$37,369.00**
Windows Server 2008 Standard (64-bit)	4	1,199.00	$4796.00
Windows Server 2008, External Connector License	4	$1,999.00	7,996.00
Subtotal Operating System Software			**12,792.00**
Total (HP BladeSystem hardware and Windows Server 2008 OS)			**$50,161.00**

Pricing for the Application Server Used in the Tests

Pricing includes middle tier software licensing costs for the primary application server used in the remote tests. Database software costs and middle tier/database hardware costs were not included, since these were not part of the system under test (SUT). We priced IBM WebSphere 7 Application Server Edition for the application server, running on both Power6 and Intel XEON quad-core CPUs. Note the WebSphere 7 Application Server Software costs 140% more per core to deploy to IBM Power6 than to Intel/AMD platforms running Windows Server 2008 (50 Processor Value Units per core for Intel/AMD vs. 120 Processor Value Units per core for Power6/IBM Power 570). Note that for the .NET 4.0/Microsoft Windows Server 2008 configuration, no separate application server is necessary: .NET is integrated into Windows Server and new versions are made available as free downloads on MSDN. There is also no redistribution license fee to redistribute the full .NET Framework runtime.

IBM WebSphere 7/IBM Power 570

Pricing Obtained from IBM Passport Advantage Published Pricing at http://www-01.ibm.com/software/lotus/passportadvantage/pvu_licensing_for_customers.html.

IBM WebSphere Application Server Processor Value Unit (PVU) License + SW Subscription & Support 12 Months (D55W8LL)……960[1] @ $46.25……$44,400.00

Total WebSphere Application Server License Costs……$44,400.00

IBM WebSphere 7/ Hewlett Packard BladeSystem C7000

Pricing Obtained from IBM Passport Advantage Published Pricing at http://www-01.ibm.com/software/lotus/passportadvantage/pvu_licensing_for_customers.html.

[1] 120 PVUs per Power6 core; 8 Power6 cores total on system as tested.

IBM WebSphere Application Server Processor Value Unit (PVU) License + SW Subscription & Support 12 Months (D55W8LL)……..800[2] @ $46.25……..$37,000.00

Total WebSphere Application Server License Costs……..$37,000.00

Total Hardware, Operating System and Application Server Costs for IBM WebSphere 7/AIX on IBM Power 570 as tested

$260,128.08

Total Hardware, Operating System and Application Server Costs for IBM WebSphere 7/Windows Server 2008 on Hewlett Packard BladeSystem C7000 as tested

$87,161.00

Total Hardware, Operating System and Application Server Costs for Microsoft .NET/Windows Server 2008 on Hewlett Packard BladeSystem C7000 as tested

$50,161.00

[2] 50 PVUs per Intel XEON core; 16 Intel Xeon cores total on system as tested.

Appendix B: Tuning Parameters

WebSphere Tuning – IBM Power 570/AIX

Servlet Caching turned on in Web Container

Session State set to 5 minute expiration (in-process session state)

Access Log Turned Off

Performance Monitor Infrastructure Turned Off

App Profile Service Off

Diagnostic Trace Turned Off

System Out Off

JVM Configured not to write System.Out printline messages

HTTP Channel maximum persistent requests = -1

Minimum Web Container threads = 50

Maximum Web Container threads = 50

Minimum ORB threads = 24

Maximum ORB threads = 24

Minimum Default threads = 20

Maximum Default threads = 20

Minimum JDBC Connections in Pool = 51

Maximum JDBC Connections in Pool = 51

Java Heap Size: 2000 min/2100 MB max

IBM HTTP Server Tuning AIX

Access Log Off

Max KeepAlive Requests unlimited

ThreadLimit 500

ServerLimit 22

StartServers 22

MaxClients 11000

MinSpareThreads 25

MaxSpareThreads 75

Threads/Child 500

MaxRequests/Child 0 (unlimited)

WebSphere Tuning – Windows Server 2008 R2

Servlet Caching turned on in Web Container

Session State set to 5 minute expiration (in-process session state)

Access Log Turned Off

Performance Monitor Infrastructure Turned Off

App Profile Service Off

Diagnostic Trace Turned Off

System Out Off

JVM Configured not to write System.Out printline messages

HTTP Channel maximum persistent requests = -1

Minimum Web Container threads = 50

Maximum Web Container threads = 50

Minimum ORB threads = 24

Maximum ORB threads = 24

Minimum Default threads = 20

Maximum Default threads = 20

Minimum JDBC Connections in Pool = 51

Maximum JDBC Connections in Pool = 51

Java Heap Size: 2000 min/2100 MB max

IBM HTTP Server Tuning Windows Server 2008 R2

Access Log Off

Max KeepAlive Requests unlimited

Default Threads 3200

Threads/Child 3200

.NET Tuning

.NET Worker Process

Rapid Fail Protection off
Pinging off
Recycle Worker Process off

ASP.NET

Forms Authentication with protection level="All"

Forms Authentication Timeout=5 minutes

IIS 7.0 Virtual Directory

Authentication Basic Only
Access Logging Off

Service Behavior for Trade Business Services:

<behavior name="TradeServiceBehaviors">
<serviceDebug httpHelpPageEnabled="true" includeExceptionDetailInFaults="true"/>
<serviceMetadata httpGetEnabled="true" httpGetUrl=""/>
<serviceThrottling maxConcurrentInstances="400" maxConcurrentCalls="400"/>
</behavior>

.NET StockTrader

Max DB Connections = 60

Min DB Connections = 60

DB2

Logging files expanded to 15 GB

Logging Set to One Drive Array (array a)

Database file on Second Drive Array (array b)

Max Application Connections = 400

SQL/Server

Logging files expanded to 15 GB

Logging Set to One Drive Array (array a)

Database file on Second Drive Array (array b)