



Mainsoft Ports a Composite Application Platform to Java EE While Maintaining Original .NET Performance

Mainsoft ports server components of an award-winning composite application platform solution to WebSphere® Application Server in three months. The ported application delivers equivalent performance and scalability under Java™ EE on Linux® as the original .NET application.

May 2007

Contents

Executive Summary	2
Introduction	2
The Mainsoft Approach	3
Porting the Application Platform	4
Measuring Performance, Scalability, and Responsiveness Under Increased Load	4
Test 1: Measuring the Throughput of the .NET and WebSphere Versions	5
Test 2: Measuring the Scalability of the .NET and WebSphere Versions	6
Test 3: Measuring Responsiveness of the .NET and WebSphere Versions Under Increasing User Load	7
Conclusion	8



Executive Summary

The process of porting applications from .NET to Java EE is typically an expensive and disruptive one. Mainsoft used its software to port a popular business integration software platform from .NET to WebSphere as a pure Java EE application. The original .NET version and the Java EE version running under Linux were then thoroughly tested in IBM's Chicago Innovation Labs under three different user scenarios, demonstrating complete preservation of performance, scalability, and response time of the original .NET version.

Introduction

The composite software platform enables rapid assembly and deployment of Service Oriented Architecture (SOA)-based applications, integrating business functionalities between disparate applications. The platform provides tooling and deployment capabilities that allow developers to rapidly integrate existing Web and software services into a new composite application using a drag-and-drop methodology to tie them together. Where Web services do not exist, the platform creates a service facade that interfaces to existing software assets and allows them to be reused.

The platform is designed to be a coherent platform that allows for existing assets to be refined and turned into business services, for services to be assembled into user-oriented applications, and for these applications to be deployed for use.

The suite includes a studio – a development environment that allows developers to refine existing services into units of business service functionality that may be assembled into a composite application. It also includes a repository in which services may be published and later discovered. The complete suite is represented by the diagram in Figure 1.

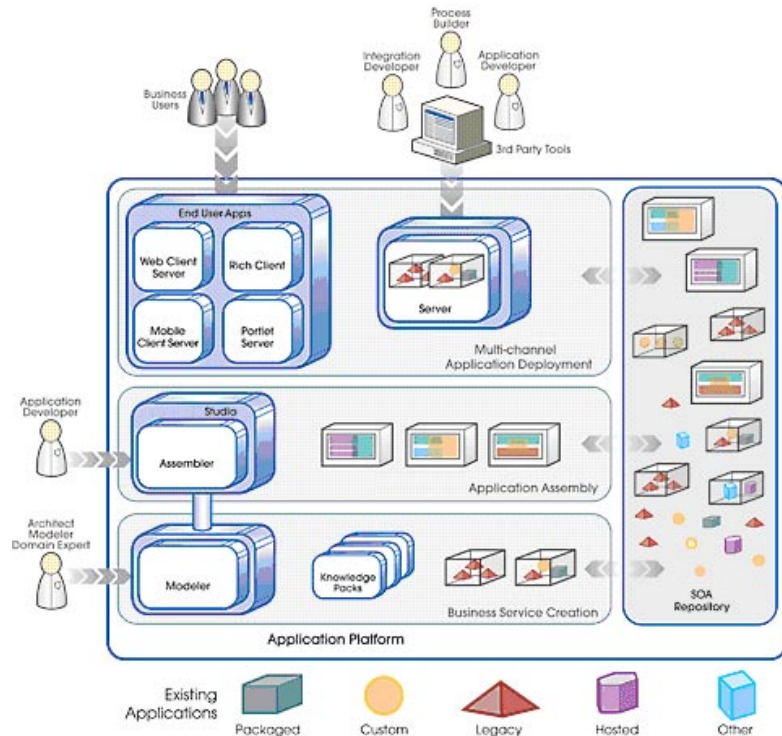


Figure 1. The application platform



The runtime environment is seen at the top left of this diagram. A number of different client access types are available to end users including a Web-based client, a Rich client, a Mobile client, and a Portlet-based client that allows service applications to be incorporated in commercial portals. Application developers and modelers use the studio to build services that can then be published into the repository and accessed by applications running on these clients.

This entire application stack was written for the Microsoft .NET platform, running on the Microsoft Windows operating system, and using C# as the primary development language.

The application platform vendor made the decision to broaden its technology to support customers that do not use Windows or .NET and to offer Java EE support for its application platform. The vendor recognized that rewriting the application, consisting of 260,000 lines of C# code in Java, so that it may run on multiple platforms, would divert critical development time from core innovation. On average, it takes 18-24 months and a full development team to rewrite a platform of this size so an alternative was needed to expand deployment options while keeping their core developers focused on product innovation.

The Mainsoft Approach

Mainsoft's cross-platform development and porting toolkit, Mainsoft, Enterprise Edition, allows companies to cross compile their applications to Java EE while preserving their existing .NET assets. Mainsoft integrates seamlessly into the Visual Studio® development environment, enabling C# and Visual Basic® developers to rapidly develop and maintain server and Web applications that run on .NET and Java EE platforms, thereby reducing application development and maintenance costs, time-to-market, and total cost of ownership.

All applications built for the .NET platform are compiled into the Microsoft Intermediate Language (MSIL), which in turn is compiled into native machine code at runtime. Mainsoft is based on a cross compiler that converts MSIL to Java bytecode. Developers can write programs in either C# or Visual Basic and use this compiler to target Java EE application servers, producing Java bytecode from their .NET-based source code, and deploying it using standard Java Web Archive (WAR) deployment technology. The runtime dependency classes from the .NET Framework are included, as Mainsoft provides a pure Java runtime implementation of the server-side .NET Framework class libraries, including ASP.NET, ADO.NET, and Web services. This is shown in Figure 2.

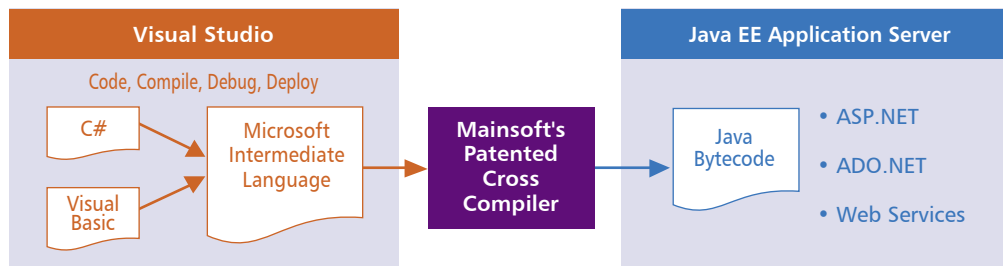


Figure 2. Mainsoft's Patented Cross Compiler

Maintaining the application as it runs on Java EE is also achieved using the existing developer skills with the Visual Studio IDE. The Mainsoft plug-in to the Visual Studio IDE has been designed to make the most of existing developer skills, so minimal retraining is necessary for building and maintaining ported applications. It is important to note that Mainsoft does not translate source code into Java, so developers can continue using the .NET language that they are familiar with. Instead it performs the translation at compile time and at the bytecode level.



Mainsoft introduces the ability to reference and consume Java components from within their C# or Visual Basic-based applications.

Additionally, the experience of maintaining an application is also consistent. Applications running on the Java EE stack may also be debugged from within the Visual Studio development environment, greatly easing root cause analysis of failures at runtime.

Porting the Application Platform

Mainsoft evaluated the source code for the application platform and delivered a plan outlining scheduling, milestones, resource allocation, and a fixed cost to complete the project. The team implemented the server deployments on Java EE with very little recoding [measured at about 0.5% of the code needing modification], using the Mainsoft technology.

Mainsoft delivered the server deployments on Java EE, certified the application as IBM Server Proven to run on IBM platforms, and also validated the application on BEA WebLogic and Apache Tomcat. Mainsoft also worked with the vendor technical staff to fine-tune the application to ensure optimal performance in its new operating environment.

Finally, Mainsoft helped the vendor to establish a single source code development environment for long term maintenance, enabling them to maintain both .NET and Java EE deployments of the application platform using a single source code base, and without any retraining in the Java language.

Mainsoft ported the application in three months, and the vendor estimates Mainsoft accelerated its development process six to nine months and saved the company more than US \$1 million versus rewriting the application from scratch.

Measuring Performance, Scalability, and Responsiveness Under Increased Load

For the application platform vendor, it was critical to deliver on identical hardware configuration, equivalent level of performance with WebSphere as with the one offered on Windows by the .NET version of the application platform.

With this in mind, Mainsoft and the vendor technical staff set up a testing environment where the .NET and the WebSphere versions of the application platform could be compared. The tests were designed to focus on three different aspects of the system – performance, scalability, and responsiveness under increased load.

The first test measured the throughput of the two versions on identical machines, measuring the number of Requests per Second (RPS) handled by each version, under constant optimal user load.

The second test measured the ability of the two versions to scale and increase their throughput with the addition of CPUs.

The third test observed how the response time of the two versions is influenced as the user load grows.



Test 1: Measuring the Throughput of the .NET and WebSphere Versions

The purpose of this test was to determine how the two versions handle a constant, heavy load of concurrent incoming requests over time.

A client system was designed to simulate 400 concurrent users that loop through two steps: sending an http request to the tested server and waiting for the server to reply. This is repeated throughout the duration of the test.

The number of the concurrent users and the rate at which they would send the requests was determined after conducting a series of tests aimed at finding an effective level of load for the two application versions. Under-loading the systems would not have reflected their true capacity, while overloading them would have introduced irrelevant results.

The load was generated by a collection of five identical IBM System x (x335, 2 x 2.8 GHz CPU, 2 GB Memory, 2 x 36 GB Disk, 1 GB Ethernet) client machines that were set up and with IBM's Rational Performance Tester. One of the client machines was designated as the dispatcher and the other four as agents.

The test was conducted for an hour. During that time, the overall number of the requests handled by the server being tested was recorded and used to calculate the average number of handled requests per seconds over time.

The same test was repeated three times, once for the .NET version, and twice for the WebSphere version, which was tested both on Windows and on Linux.

In all three tests, the server machine was identical – an IBM System x (x366, 4 x 3.6 GHz CPU, hyperthreaded, non-dual core, 8 GB Memory, 2 x 73 GB Disk RAID1, 1 GB Ethernet).

Figure 3 shows the average number of requests per second in this scenario measured for 60 minutes. As can be seen, the WebSphere versions of the application performed very favorably compared to the original .NET version, handling around 8% more requests per second across all user loads.

The steep decrease in RPS on the Linux WebSphere version around minute 42 can be explained by garbage collection activity taking place at that time.

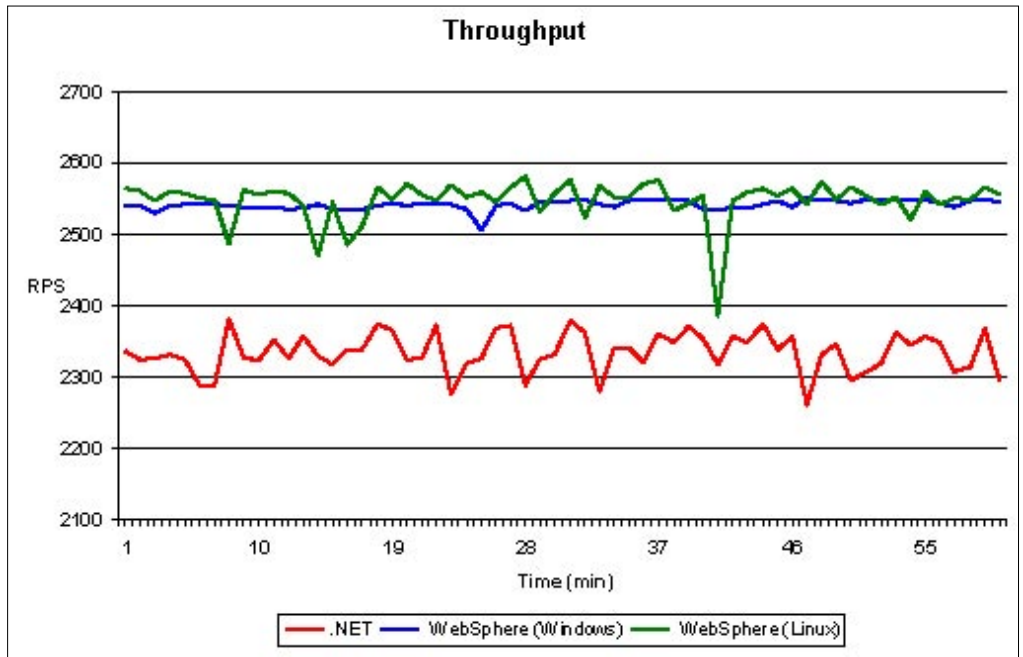


Figure 3. Throughput for application



Test 2: Measuring the Scalability of the .NET and WebSphere Versions

Another set of tests was performed, exploring the ability of the two application versions to increase their throughput as more CPUs are added to the system.

These tests used the same 4-CPU hardware as in the previous test, but this time, the tests were repeated four times for each of the server versions. With the first run, only one of the four CPUs was enabled. Two CPUs were enabled on the second run, three on the third run, and lastly – four CPUs were enabled on the fourth run.

To ensure that the systems are kept busy at optimal rate at all runs, the client system was set to simulate 100 concurrent users for the first run, 200 for the second, 300 for the third, and 400 for the fourth run.

The average requests handled per second was measured in each of these runs for both the .NET version and the WebSphere version, and the results can be seen in Figure 4.

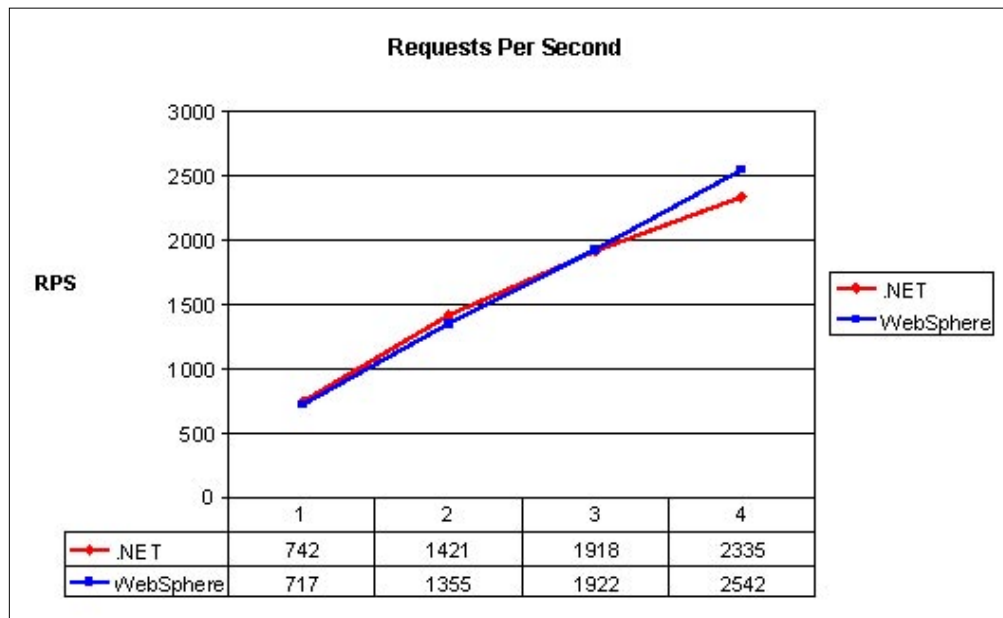


Figure 4. Average requests per second in 1,2,3,4 CPU systems

In single and dual CPU systems, the performance is very close, with the .NET being a little more efficient than its Java-ported counterpart. Once a larger hardware configuration is employed, the scalability aspects of WAS begin to show, with a noticeable improvement in the number of requests per second that it can perform (2542 against 2335) in a 4-CPU system.

In addition to these tests, a similar test was run on an AIX system (System p 570+, 8 x 2.2 GHz Power5+, 32GB Memory, 6 x 70 GB Disk, 1 GB Ethernet), with 125 concurrent users per CPU. This test could not be duplicated for the .NET version of the application platform as the .NET version can not execute on such a system. This additional test measures the return on hardware investment when adding more CPU to an AIX machine. The test was run with 250 users on a 2-CPU system, 500 users on a 4-CPU system, etc. The results can be seen in Figure 5.

The ability to run the application as a Java EE one means that it can be run on AIX, enjoying a close to linear scalability on that system.

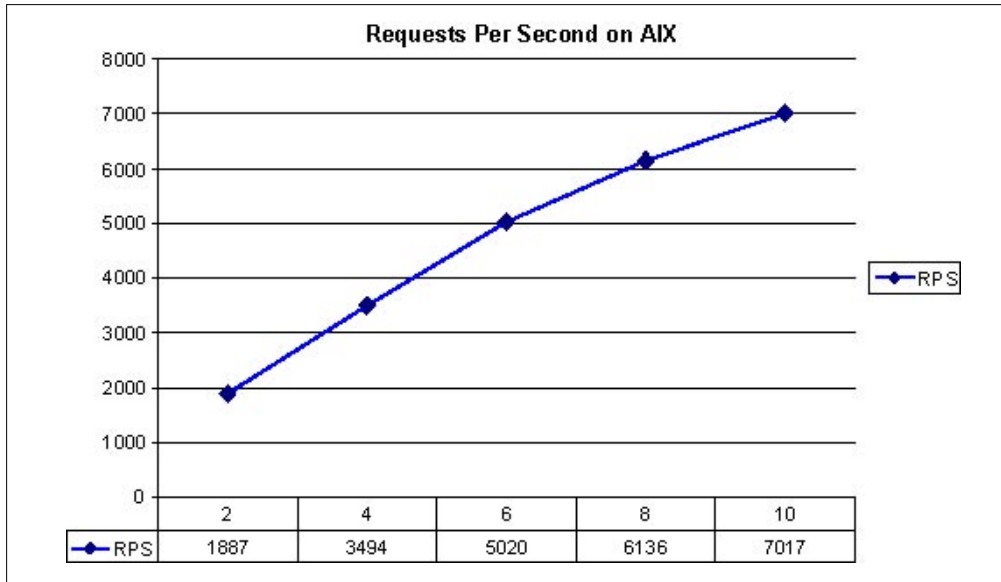


Figure 5. AIX scalability results

Test 3: Measuring Responsiveness of the .NET and WebSphere Versions Under Increasing User Load

The application was further tested on .NET and WebSphere on identical 4-CPU systems. The same test was repeated for each of the application versions with different user load profiles. The first run involved using 1000 concurrent users, and subsequent runs increased this by 1000 users each time until the final run of 4000 users was performed.

It was generally expected that as the load on the server increases, the average response time per single request be longer.

The purpose of the test was to observe the rate at which the two server versions' response time deteriorates, as the client load is increased.

Figure 6 shows how the average response time gets longer as the load increases from 1000 through 4000 concurrent users.

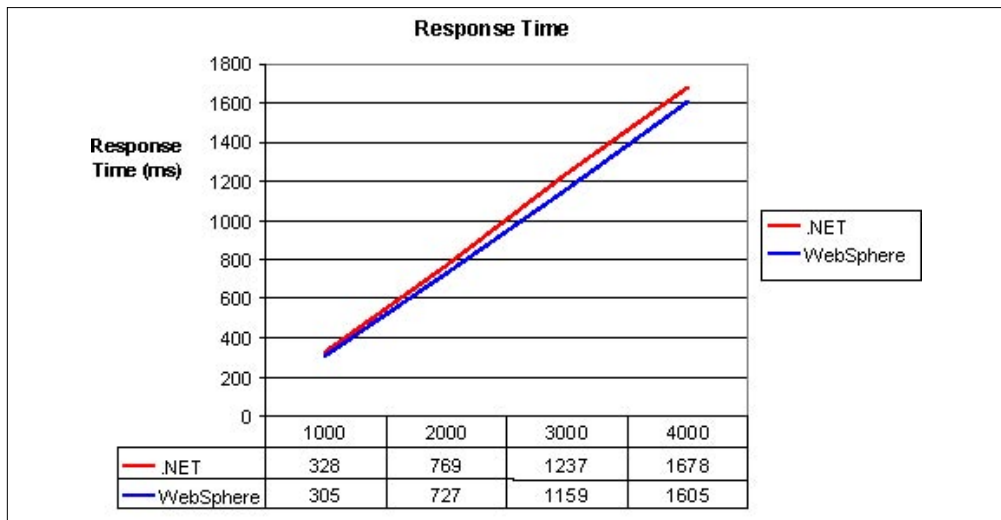


Figure 6. Response time across different user loads



As can be seen in the graph in Figure 6, the ported version of the application is more responsive compared to the original .NET version, providing at least 4% faster response time across all user loads.

Figure 7 shows how the throughput of the two versions is influenced by the added load.

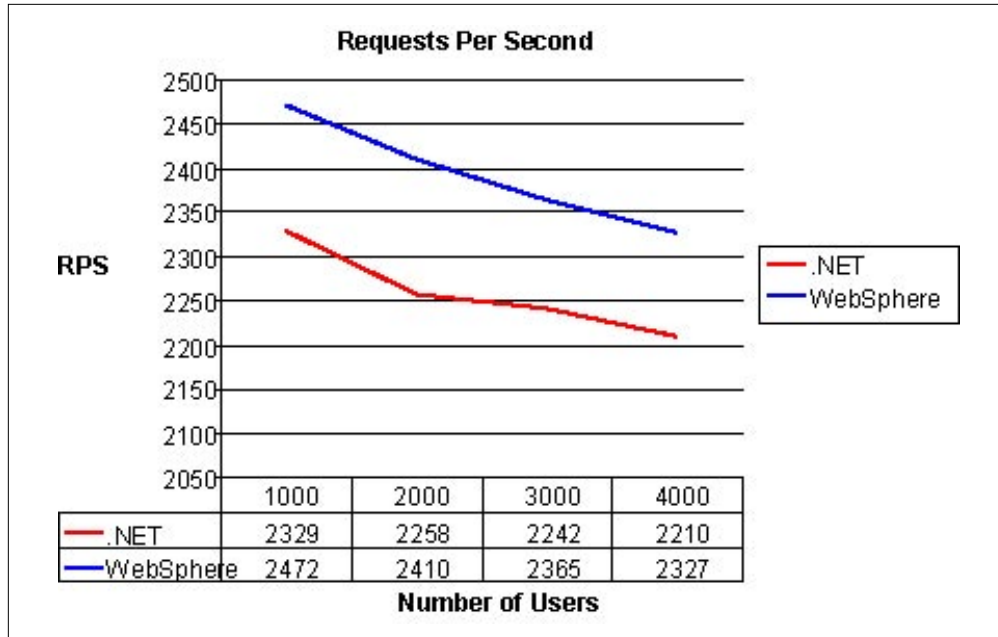


Figure 7. Requests per second across different user loads

It can be seen that the throughput gradually decreases as the user load increases. This can be expected, as having more load means consuming more memory and system resources. A similar gradual decrease is seen in both .NET and WAS.

It can be deduced from these results that the porting process does not introduce a performance overhead. Moreover, on a WebSphere Application Server, the performance of the application can be seen to be slightly better.

Most importantly, if a Service Level Agreement (SLA) needs to be met, it has been demonstrated that the same application can run on equivalent hardware, and is able to meet the same SLA.

Conclusion

The software vendor successfully ported their .NET based application platform onto Java EE using Mainsoft's technology and professional services saving \$1M in R&D costs. The performance of this application was tested on the same hardware and under the same stress test conditions. The ported version of the application showed equivalent performance in average response time, and equivalent performance in server throughput. Thus, an SLA could be met using the ported version of the application without additional hardware investment. The Java EE version of the application also showed that in multi-CPU environments, the scalability characteristics of Java EE and WebSphere Application Server began to show marked improvements over the original .NET application.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.