Web Services Performance

Comparing Java^{TM} 2 Enterprise Edition (J2EE^{TM} platform) and .NET Framework

Sun Microsystems Inc.

June 2004

Introduction

Web services are Web-based enterprise applications that use open, XML-based standards and transport protocols to exchange data with calling clients. Web services are becoming the most important technology for communications between disparate applications in different enterprises. Even within an enterprise, web services are fast becoming a de facto standard as more applications use XML to communicate and store data.

As web services deployments go mainstream, it is important for organizations to understand the features and performance of the implementations of this technology in various product offerings.

In this paper, we consider the performance of web service technologies in the two primary middleware platforms today: J2EE and .NET. Both, the J2EE platform and .NET framework offer similar facilities for creating and using web services. We consider a basic web service method that is called with different types and sizes of arguments in both platforms and compare its performance.

Summary of Findings

In all the tested cases, the J2EE platform outperformed .NET. We took care to ensure that the application code used on both platforms are as similar as possible in order to make an apples-to-apples performance comparison. We highlight the results in the sections to follow and also provide the reader with the information necessary to conduct the test independently and confirm the superior web services performance of J2EE technology.

Test Description

WSTest is a web services test developed at Sun Microsystems. It is designed to measure the performance of various types of web services calls, which are described below :

- **echoVoid** – Sends and receives an empty message. This tests the performance of

the web services infrastructure.
- **echoStruct** – Sends and receives an array of size 20 – each element is a structure composed of one element each of an integer, float and string data type.
- **echoList** – Sends and receives a linked list of 20 elements – each element is a Struct as defined in echoStruct.
- **echoSynthetic** – Sends and receives multiple parameters of different types – String, Struct and a byte array of size 1K.

WSTest simulates a multi-threaded server program that makes multiple web services calls in parallel. To avoid the effects of other platform components, the web service methods perform no business logic but simply return the parameters that were passed in.

WSTest measures the throughput of a system handling multiple types of web service requests. The notion of a web service operation here corresponds to a request/response cycle. WSTest reports the **Throughput** - Average number of web service operations executed per second and the **Response Time** – Average time taken to process a request. These metrics are reported for each of the 4 types of operations tested.

Test Details

WSTest can be configured using the following parameters, specified in an initialization file:

- **Agents** - This is the number of client threads and is set to maximize CPU utilization and system throughput.
- **RampUp** - Time allotted for warmup of the system.
- **SteadyState** - The interval when throughput is measured.
- **RampDown** - Time allotted for rampdown, completing operations in flight.
- **EchoVoidMix** - %Mix of operations that are EchoVoid
- **EchoStructMix** - %Mix of operations that are EchoStruct
- **EchoListMix** - %Mix of operations that are EchoList
- **ListSize** – size of list for echoList
- **EchoSyntheticMix** - %Mix of operations that are EchoSynthetic
- **NumBytes** – size of byte array for echoSynthetic

WSTest reads these properties at initialization into an in-memory structure that is then accessed by each thread to initiate an operation as per the defined mix. A new operation is started as soon as a prior operation is completed (there is no think time). The number of operations executed and the response time is accumulated during the SteadyState period and is reported at the end of the run.

System Configuration

WSTest was run on the following system configuration (the same hardware and Windows operating system was used for both, J2EE and .NET platforms). The client drivers were run on a different system with the same configuration. Both the systems were directly connected using a 1GB ethernet link.

- Dell Server 4600
- 2 Intel Xeon cpus at 2.6 GHz (2-way hyper-threaded)
- 2 GB main memory
- Windows Server 2003 Standard Edition
- J2SE 1.4.2 SDK (uses 1 GB heap)
- JWSDP 1.3

The Java Web Services Developer Pack (JWSDP) Version 1.3 was used for testing the JAX-RPC implementation. The pack includes Tomcat 5.0 as the web server. Windows Server 2003 includes .NET 1.1 and IIS 6.0 as the web server.

WSTest Configuration

For the test results reported, WSTest was run with the following parameters set in the initialization file, with the mix changed to 100% for each of the types :

```
#Agents determines the number of concurrent threads
Agents          = 8
#Rampup time for the run in seconds
RampUp          = 300
#SteadyState time for the run in seconds
SteadyState       = 300
#Rampdown time for the run in seconds
RampDown       = 10
#Output directory where the results are generated
OutputDir       = ../../results
#Prefix for the output result file
OutputFilePrefix    = WSTestReportSummary
# %Mix of the different types of web services calls
EchoVoidMix       = 100
EchoStructMix       = 0
EchoListMix       = 0
EchoSyntheticMix    = 0
NumBytes       = 1024
ListSize       = 20
```

Windows Tuning

For both the J2EE platform and .NET runs, the server was tuned as follows :

- The following services were disabled, as this is a dedicated web server :
  Alerter
  ClipBook
  Computer Browser
  DHCP Client
  DHCP Server
  Fax Service
  File Replication
  Infrared Monitor
  Internet Connection Sharing
  Messenger
  NetMeeting Remote Desktop Sharing
  Network DDE
  Network DDE DSDM
  NWLink NetBIOS
  NWLink IPX/SPX
  Print Spooler
  TCP/IP NetBIOS Helper Service
  Telephony
  Telnet
  Uninterruptible Power Supply

- The following parameters were changed in
  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
  - TcpMaxSendFree - Changed to 0xFFFF, from 5000, to bump the size of the TCP Header table
  - TcpWindowSize - Changed to 64240 (maximum size)

.NET Tuning

- The Garbage Collection mode was modified to run in server mode.

IIS Tuning

- Logging was disabled.

The Application Pool was tuned as follows :
- The worker processes were prevented from being shutdown.
- The number of requests to the kernel request queue was not limited.
- The number of worker processes in the web garden was set to 4.
- Pinging and rapid-fail protection was disabled.

In web.config, the following changes were made for the application's website :
- session state management was disabled.
- authentication mode was set to None.

- Debug for compilation was set to false.

J2EE Tuning

1. The following parameters were changed from their default value for Tomcat in <JWSDP_HOME>\conf\server.xml :

```
<!--  Disable access log writing
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
prefix="access_log." suffix=".txt" resolveHosts="false"/>
-->

<!-- Disable Connector lookups of Non-SSL connections
<enableLookups="false"/>
-->

<!-- Set the minimium processors of Non-SSL connections
<minProcessors="8"/>
```

2. The JVM options for Catalina server in <JWSDP_HOME\jwsdp-shared\bin\launcher.xml were set as follows :

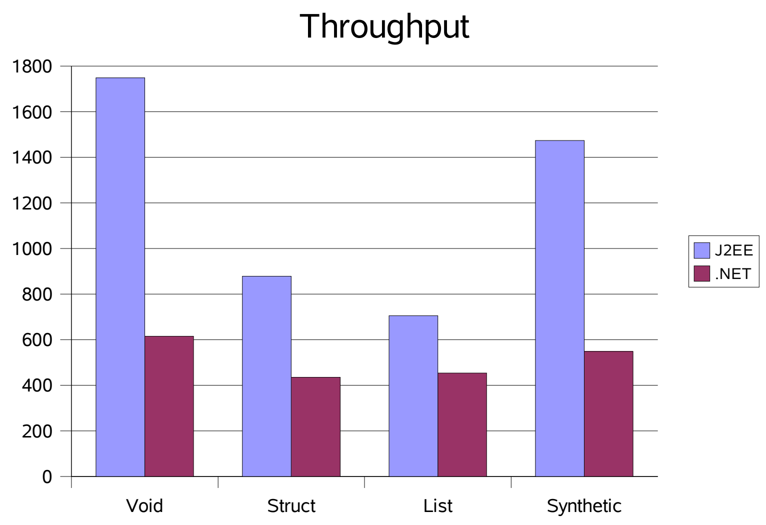
```
<launch classname="org.apache.catalina.startup.Bootstrap" ... >
<jvmarg value="-server" />
<jvmarg value="-Xms256m" />
<jvmarg value="-Xmx256m" />
```

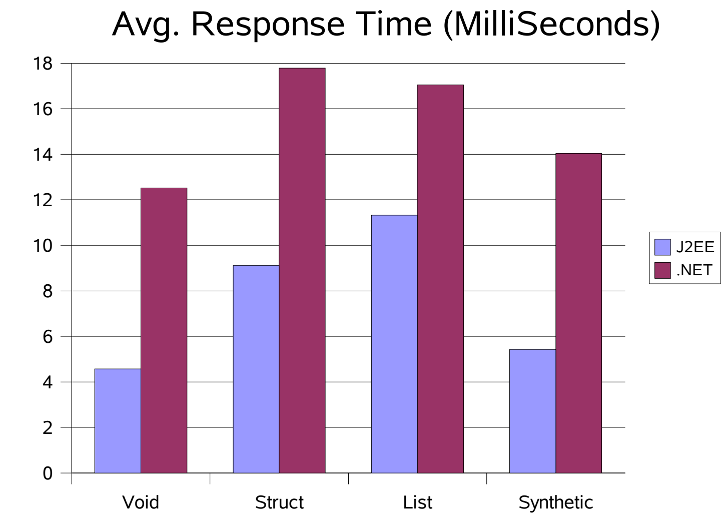
3. On the client side, the JVM options were changed in <WSTest_HOME>\Java\src\build.xml :

```
-Dhttp.maxConnections=256 -Xms300M -Xmx300M
```

Performance Results

The measured throughput and response times obtained are shown below. In the basic web services call, echoVoid, and the most complex one, echoSynthetic, JAX-RPC performs nearly 3 times better than .NET. In the other cases, J2EE technology performs nearly twice as well as .NET.





Conclusion

Web services are becoming the most important technology for communications between disparate applications in different enterprises or even within the enterprise itself. The performance of core SOAP-based web services calls is therefore crucial and must be taken into consideration when evaluating web services platforms. In this paper, we used WSTest to compare the performance of the J2EE and Windows .NET platforms when performing basic web services. Our findings indicate that for almost all meaningful cases, the JAX-RPC based web services technology of the J2EE platform offers better performance and scalability than .NET. Moreover, since the J2EE platform is completely portable, developers can expect to see this top-of-the-line performance on the Linux and Solaris platforms as well.