Developing High Performance Web Services using Apache Axis

Philadelphia Java Users Group
August 18, 2004
Overview

- Introduction
- Web Services Overview
  - Definitions, Benefits
- Developing Web Services
  - Platforms, Interoperability, Security, Browser Integration
  - Performance Considerations
- Performance Examination of SOAP using Tomcat/Axis
  - Performance Testing Methodology
  - Binary Data Transmission
  - Structured Data Transmission
- Conclusions, Links, References
Presenter - Ted O’Connor

- Senior Software Engineer at Wingspan Technology, Inc. in Blue Bell, PA
- Developers of DocWay Server – a high performance Web Services interface to Documentum
- Presentation based on recent experiences
- Been working with Java since 1.0 in 1996
- Focus is on enterprise portal and database integration using web services
- Sun Certified Java Developer
- Former Villanova ACM member and officer
Introduction

• What are Web Services?
• Why are Web Services important?
• Key features of high-performing Web Services
• Apache platform for Web Services
Corporate Web Service Adoption

- 30.5% Evaluating
- 14.2% Running a pilot project
- 14.2% Completed one project
- 23.9% Completed multiple projects
- 9.6% Not sure
- 7.6% No plans

Base: 197 IT managers at U.S. companies
Source: COMPUTERWORLD, 08/16/2004, p.41
Primary Web Service Drivers

1. Solve internal integration issues
2. Solve external integration issues
3. Extend legacy systems
4. Reduce development and maintenance time
5. Deploy applications at less cost
6. Create reusable modular code

Base: 154 IT managers involved in Web services projects
Source: COMPUTERWORLD, 08/16/2004, p.41
Top Adoption Headaches

- Security: 50.6%
- Funding: 50.0%
- Difficulty of re-engineering existing apps: 43.5%
- Risk of using for mission-critical systems: 38.3%
- Amount of new technology needed: 37.7%
- Prioritizing this work with IT initiatives: 37.0%
- Finding or training skilled IT staff: 35.7%

Base: 154 IT managers involved in Web services projects; multiple responses allowed
Source: COMPUTERWORLD, 08/16/2004, p.41
Web Services Overview

- Definitions
- Benefits
- When to use Web Services
- Common Web Services Platforms
Web Services Definitions

• SOAP
  – A lightweight XML-based protocol for representing a remote procedure call request/response sequence.

• WSDL (Web Services Description Language)
  – An XML format for describing the API exposed by a particular web service

• UDDI (Universal Description, Discovery and Integration)
  – A mechanism for cataloging and identifying available web services

• WS-Security
  – A Web Services extension for applying security elements such (encryption, digital signatures, etc…) to individual elements within a SOAP message

• WS-Interoperability
  – Sets of Web Service specifications that define how SOAP should be used to maximize interoperability across implementations.
Web Services Benefits

- **Platform and Language Neutral**
  - Web services can be implemented and consumed using languages such as Java, C/C++, C# and more

- **Transport Neutral**
  - Web services can be invoked over virtually any network protocol including HTTP, HTTPS, SMTP and Message Queues

- **Industry-standard**
  - SOAP 1.1 is a World Wide Web Consortium Standard
  - Backed by many key technology organizations (Microsoft, IBM, Apache)
When to Use Web Services

- Proprietary or legacy functionality needs to be exposed to the widest possible audience
- Existing applications and frameworks must be preserved
- Industry-standard protocols are important
- Minimize installation and maintenance
- Multiple languages, platforms and programming environments are involved
- When you’re just not sure what your environment will look like in the future
- *The decision to use Web Services is often a strategic one, not a tactical one*
Developing Web Services

- Platforms
- Interoperability
- Security
- Browser Integration
- Performance Considerations
  - Platform/Environment
  - Binary Data
  - Structured Data
  - Session Management
  - Clustering
Web Services Platforms (most common)

- Apache Tomcat/Axis
- BEA WebLogic
- IBM WebSphere
- Microsoft .NET
- Systinet WASP
Web Services Interoperability

• WS-I Standard was created to help resolve lingering compatibility issues
• It does not change the SOAP standard it just defines how it should be used to ensure interoperability.
• Compatibility is the key focus of Apache Axis 1.2
Web Services Security

• HTTPS (SSL)
  – Covers the widest range of security considerations
  – Easily applied to any SOAP implementation
  – Performance considerations

• WS-Security
  – Far fewer performance implications
  – Apply desired levels of security individually to components of a SOAP message
  – Requires support within the SOAP implementation
  – Currently in the early stages of industry-wide adoption

• Other options
  – Not widely adopted throughout the industry
Web Services Browser Integration

- Web Applications
  - Java Servlet, JSP, ASP.NET, etc…

- Portal Frameworks
  - JSR 168, WSRP, Proprietary Portal Frameworks

- Content Transfer
  - Convert between SOAP and HTTP
Web Services Performance Considerations

- SOAP Implementation
  - SOAP performance can vary greatly depending on the SOAP implementation being used.
  - Be aware of compatibility issues between different SOAP implementations and their corresponding SOAP Clients.

- Application Server

- Java Virtual Machine
  - Pay careful attention to settings and startup flags
  - Use the `–server` option with Sun JVMs
Web Services Performance Considerations

• Exposed API
  – Primitive types (integer, string, float) are handled much faster than custom types.
  – Consider using XML or a similar alternative to represent complex structures.

• Transmitting Binary Data
  – Can be embedded directly in a SOAP message
  – Can also be attached outside the SOAP envelope
    • DIME is slightly faster than MIME, and more widely supported
Web Services Performance Considerations

- Clustering
  - SOAP is designed to work over stateless protocols such as HTTP. It is often best to design a similarly stateless API.
  - Stateless Web Services can be easily clustered when it becomes necessary to scale beyond a single server machine.
  - A simple random or round-robining brokering scheme is usually sufficient.
  - Load-balancing hardware can also be used
Session Management

- Depending on the application you should either:
  - Expose a session mechanism through its API (Not recommended)
  - Accept user identification and/or credentials with each invocation
  - Execute all requests as a well-defined user

- Minimize the number of requests
  - SOAP overhead is about as high as you can get for an RPC mechanism

- Build a stateless API
Performance Examination of SOAP

• Methodology
• Examination of Specific Issues
  – Binary Data Transmission
  – Structured Data Transmission
• Areas for Future Examination
• Results Verification
Methodology - Goals

- Repeatable measurements
- Easily controlled environment
- Automated testing
- Highly configurable test framework
- Baseline best case scenario
Methodology – Testing Technique

• Pseudo Scientific Method

• Hypothesis or Query
  – Select an area for examination

• Experiment
  – Perform a sequence of test runs keeping all parameters constant but one

• Conclusions
  – Draw conclusions from the performance results

• Lather, Rinse, Repeat.
  – Select new areas for examination based on results and conclusions
Methodology - Environment

- **Server**
  - Athlon XP 2600+
  - 1 GB RAM
  - 80 GB IDE 7200 RPM HD
- **Test Clients**
  - Pentium 3 650-850 MHz
  - 512 MB RAM
- **Software**
  - Apache Tomcat 5.0.25
  - Apache Axis 1.1
    - Java Activation Framework 1.0.2
    - JavaMail API 1.3.1
    - XML Security 1.1.0
  - Apache JMeter 2.0.1
Methodology - Configuration

- Performance analysis project contains binary distributions of all software used (except ant)
- CVS is used to synchronize project directories across machines
- Ant tasks are used to perform all operations
  - Install and configure all software
  - Compile and deploy the web service
  - Generate client classes using WSDL2Java
  - Compile and deploy JMeter classes
  - Run tests
- Project builds and deploys both client and server pieces on each machine
Methodology – JMeter Sampler Options

- A Sampler in JMeter performs a single action and records timing information
- JMeter 2.0 supports two SOAP related samplers
  - SOAP/XML-RPC Request
  - WebService(SOAP) Request (Beta Code)
- Neither is ready for prime-time
  - Usability issues
  - Performance overhead
- General-purpose Java sampler provides fastest route to developing a custom sampler to JMeter
Methodology – JMeter Sampler Classes

• A plug-in sampler must
  – Extend AbstractJavaSamplerClient
  – Implement Serializable
  – Override the runTest method

• Samplers can accept configuration parameters
  – Override getDefaultParameters method
  – Parameters will be presented through the JMeter GUI
  – JavaSamplerContext object contains runtime parameter values
Methodology – JMeter Tests

• Generate a client for our Web Service using WSDL2Java
• For each Web Service method, create a Java Sampler class for JMeter
• For each Sampler class, abstract the arguments for the Web Service method as Sampler parameters so they can be set by the Ant tasks
Methodology – Baseline Web Service Methods

• void sleep(int mSecs)
  – Calls Thread.sleep(mSecs) and returns
  – Returns immediately if mSecs == 0

• String echo(String msg)
  – Returns the input string
Binary Data Transfer Tests

- `byte[] produce(int numBytes)`
  - Returns a populated array of bytes of the requested size
- `void produceAttachment(int numBytes)`
  - Returns a populated SOAP attachment of the requested size
- A JMeter test class was created for each of these which was then run repeatedly with varying number of bytes and threads.
Binary Data Transfer Results

- Transmission time of Base64 vs. SOAP attachments as a function of data size
Binary Data Transfer Results

- Transmission time of 12KB through Base64 and SOAP attachments as a function of the number of simultaneous users
Binary Data Transfer Results

- Transmission time of 1KB through Base64 and SOAP attachments as a function of the number of simultaneous users
Binary Data Transfer Summary

- Base64 encoding and SOAP Attachments are almost identical in terms of scalability.
- Base64 encoding performs better with data sizes below 12KB.
- SOAP Attachments perform better with data sizes over 12KB, but become unusable once the sizes get over 75KB.
- Base64 encoding is the recommended approach since it is more reliable, generally better performing, and also easier to code.
Structured Data Transfer Tests

• Bean[] getNormalBeans(int numBeans)
  – Returns the specified number of bean objects

• String getStringBeans(int numBeans)
  – Returns the specified number of bean objects as an XML fragment which the client must deserialize back to Bean[].

• Bean[] getCustomBeans(int numBeans)
  – Calls getStringBeans() but the deserialization is performed in the client stub class.
  – Java clients see an API similar to getNormalBeans() but get the performance of getStringBeans().

• A Bean contains a getter/setter for each of five properties:
  – boolean, int, long, float, String
Structured Data Transfer Results

- Transmission time for sending structured data using three mechanisms (Default Serialization, “Custom Serialization”, WSDL2Java modified client) as a function of the number of elements being sent.
Structured Data Transfer Results

- Transmission time for 100 elements as a function of the number of simultaneous users
Structured Data Transfer Summary

• When your focus is on performance and Java clients you are better off using your own Object to XML serialization than relying on the Axis default.

• With a minor modification to the WSDL2Java generated client stub you can get a major performance improvement without changing the client API.

• Clients that don’t use the generated stub however would have to duplicate the deserialization code.

• This should be similar to the difference between SOAP RPC encoding and document/literal encoding.
Areas for Further Examination

• JVM Impact on SOAP Performance
• Axis vs. alternatives (BEA, Systinet, etc…)
• Other SOAP Encoding types (doc/lit, message, etc…)
  – See Dennis Sosnoski’s “Cleaning Up SOAP Web Services” for more information on this.
• Axis 1.2 performance benefits over Axis 1.1
• Large SOAP Attachments
• Compression
Conclusions

- Web Services are fundamentally an integration technology first.
- Web Service standards are still in their infancy. WS-Security and WS-Interoperability should go a long way to making Web Services even more practical in enterprise applications.
- Your choice of binary and structured data handling mechanisms could have as much impact on your Web Service performance as your choice of SOAP engine and application server.
- We have just seen the tip of the iceberg in terms of performance improvements, but the test suite should be able to validate most opportunities with few changes.
Results Verification

- Test project will be made available for download in the near future
- Email toconn@wingspantech.com to be notified when the project is available
Software links

- Apache Tomcat
  - http://jakarta.apache.org/tomcat
  - Current version 5.0.27
- Apache Axis
  - http://ws.apache.org/axis
  - Current version 1.1 (June 16, 2003)
- Apache JMeter
  - http://jakarta.apache.org/jmeter
  - Current version 2.0.1
Other References

- XML Protocol Working Group – W3C
  - http://www.w3c.org/2000/xp/Group/
- SOA and Web Services – IBM
- WS-Interoperability Basic Profile 1.0 Final
  - http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html
- Cleaning Up SOAP Web Services – Dennis Sosnoski (S³i)
  - http://www.sosnoski.com/presents/cleansoap/
- Performance Test Solution – PushToTest
  - http://www.pushtotest.com/Services/performancetest.html
- Squeezing SOAP – Brian Goodman (IBM)
Contact Information

Ted O’Connor
toconn@wingspantech.com
215-619-2400 x.2427