



Web Services Advanced Topics: Reliability, Interoperability, Performance

Santiago Pericas-Geertsen
Sun Microsystems

Santiago.PericasGeertsen@sun.com



**UNLOCK
OPPORTUNITY**

What will you open?

SUN TECH DAYS 2006-2007
A Worldwide Developer Conference

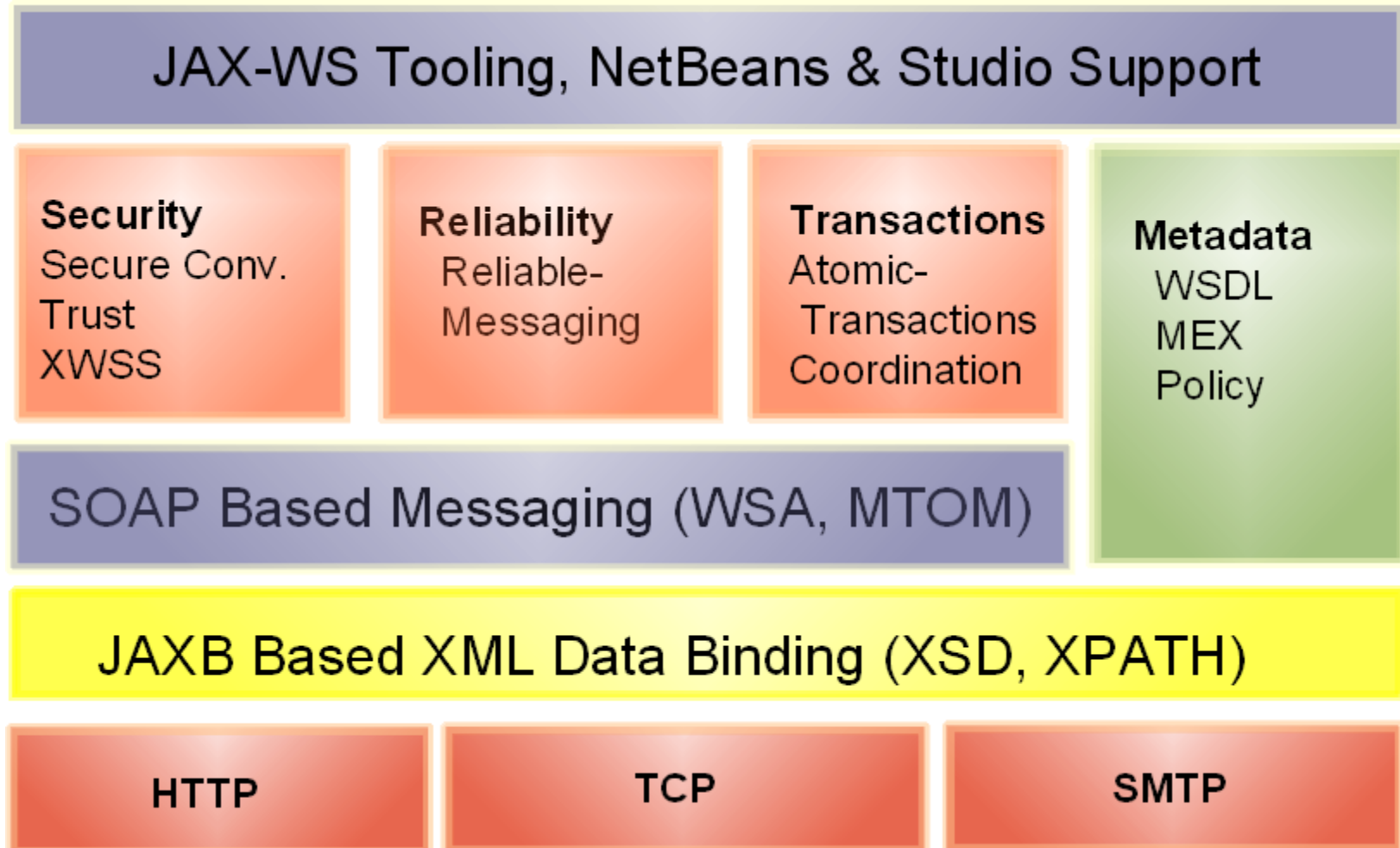
About the speaker

- Ph.D. in Computer Science from Boston University
- Staff Engineer at Sun for 4.5 years
- Participated in the following projects:
 - > XSLTC, Fast Infoset / Fast Schema, JAX-RPC, JAX-WS, WS Performance, JAXP
 - > Currently tech lead for JAXP

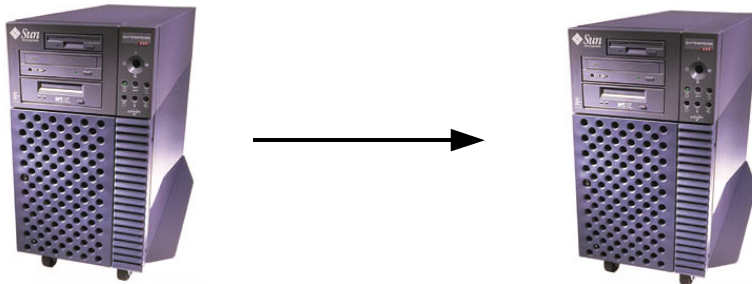
What is WSIT?

- **W**eb **S**ervices **I**nteroperability **T**echnologies
 - > A.k.a. Project “Tango”
- **G**oals:
 - > Interoperability with **M**icrosoft **C**ommunication **F**oundation (MCF, a.k.a. “Indigo”)
 - > Implementation of WS-* specifications
 - > Foundation for building SOA

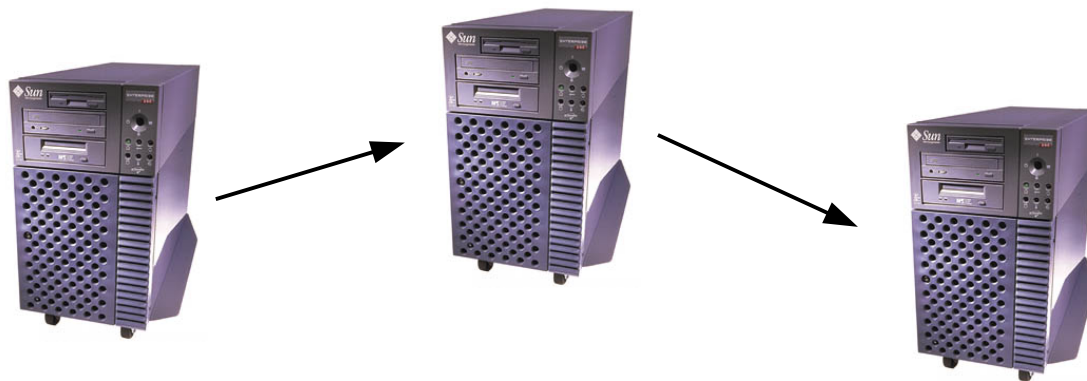
Web Services Stack



P2P vs. MP Challenges



- Addressing
- Reliability
- Security
- Transactions



Takeaways from this Talk

- How do I address a WS?
- How do I secure my messages?
- How do I know a message has arrived?
- How do I improve encoding performance?
- How do I ensure a transaction has completed?

Takeaways from this Talk

- **How do I address a WS?**
- How do I secure my messages?
- How do I know a message has arrived?
- How do I improve encoding performance?
- How do I ensure a transaction has completed?

WS-Addressing (W3C)

- Before WS-Addressing:
 - > HTTP uses URIs
 - > Not transport independent!
- Bring addressing to protocol (SOAP) layer
- Use SOAP headers to convey addressing information
- Make messages more self contained
 - > Enable use of multiple transports
 - > Go beyond request-response pattern

WS-Addressing (Contd.)

- Message addressing properties:
 - > Message ID to link request and response
 - > Destination address (where)
 - > Action to be carried out (what)
 - > Destination for replies
 - > Destination for faults
- Endpoint references:
 - > Address
 - > ReferenceParameters

Enabling WS-Addressing

- Using policy assertions in WSDL

```
<wsp:Policy wsu:Id="...">  
  <wsp:ExactlyOne><wsp:All>  
    <wsaw:UsingAddressing/>  
  </wsp:All></wsp:ExactlyOne>  
</wsp:Policy>
```

- Using Netbeans 5.5 with support for WSIT
- Using new annotations in JAX-WS 2.1

WS-A Sample Message

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <S:Header>
    <wsa:MessageID>
      uuid:6B29FC40-CA47-1067-B31D-00DD010662DA
    </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://business456.example/client1</>
    </wsa:ReplyTo>
    <wsa:To>http://fabrikam123.example/Purchasing</wsa:To>
    <wsa:Action>http://fabrikam123.example/SubmitPO</>
  </S:Header>
  <S:Body> ... </S:Envelope>
```

Takeaways from this Talk

- How do I address a WS?
- **How do I secure my messages?**
- How do I know a message has arrived?
- How do I improve encoding performance?
- How do I ensure a transaction has completed?

WS-Security (Oasis)

- Before WS-Security:
 - > SSL/HTTPS
 - > Security at the transport layer
 - > All or nothing granularity
 - > Inherently P2P
- Bring security to SOAP (protocol) layer
 - > Message security instead of transport security
- Very fine granularity using ids and Xpath
- Make messages more self contained
 - > Enable use of multiple transports

WS-Security (Contd.)

- Support for integrity, confidentiality and authentication
- Leverage existing XML recommendations
 - > XML Signature (W3C)
 - > XML Encryption (W3C)
- Support for **partial** integrity and confidentiality
 - > Only encrypt credit card number
 - > Only sign subtree rooted at element X

Enabling WS-Security

- Using policy assertions (sign SOAP body):

```
<sp:SignedParts>  
  <sp:Body/>  
</sp:SignedParts>
```

- Using Netbeans 5.5 with support for WSIT

WS-Security Sample Message

```

<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/..."
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/...">
  <S:Header>
    <wsse:Security>
      <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="..."/>
        <ds:SignatureMethod Algorithm="..."/>
        <ds:Reference URI="#XWSSGID-1">...</ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>nxUaJYA0C4x...</ds:SignatureValue>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="XWSSGID-1"> ... </S:Envelope>
  
```

Takeaways from this Talk

- How do I address a WS?
- How do I secure my messages?
- **How do I know a message has arrived?**
- How do I improve encoding performance?
- How do I ensure a transaction has completed?

WS-ReliableMessaging

- Before WS-ReliableMessaging:
 - > Reliable protocols based on TCP
 - > Inherently P2P
- Bring reliability to SOAP (protocol) layer
- At least once, at most once, in order guarantees
- Make messages more self contained
 - > Enable use of multiple transports

WS-ReliableMessaging Agents

- RM Source (RMS)
 - > Creates/terminates sequences
 - > Adds RM headers
 - > Resends messages if necessary
- RM Destination (RMD)
 - > Responds to create/terminate sequences
 - > Acks messages
 - > Discards duplicates (optionally)
 - > Ensures message ordering (optionally)

Enabling WS-ReliableMessaging

- Using policy assertions:

```
<wsaw:UsingAddressing/>  
<wsrm:RMAssertion>  
  <wsrm:InactivityTimeout  
    Milliseconds="600000"/>  
</wsrm:RMAssertion>  
<sunrm:Ordered/>
```

- Using Netbeans 5.5 with support for WSIT

WS-RM Sample Message

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:rm="http://schemas.xmlsoap.org/ws/2005/02/rm">
  <S:Header>
    <rm:Sequence>
      <rm:Identifier>uuid:00f6e634-f634-...</rm:Identifier>
      <rm:MessageNumber>1</rm:MessageNumber>
    </rm:Sequence>
    <rm:AckRequested ...>
      <rm:Identifier>uuid:00f6e634-f634-...</rm:Identifier>
    </rm:AckRequested>
    <!-- WS-Addressing headers -->
  </S:Header>
  <S:Body> ... </S:Envelope>
```

Takeaways from this Talk

- How do I address a WS?
- How do I secure my messages?
- How do I know a message has arrived?
- **How do I improve encoding performance?**
- How do I ensure a transaction has completed?

MTOM (W3C) and Fast Infoset (ITU-T/ISO)

- Before MTOM and FI:
 - > All encoded in XML
 - > Must use base64 encoding for binary payloads
 - > Not as **efficient** as pre-existing binary encodings
 - > Not as **compact** as pre-existing binary encodings
 - > Gzip = better compactness + worse efficiency

MTOM (W3C)

- Solution for large binary payloads
- Avoids base64 encoding
 - > Wraps XML message in MIME package
 - > Uses MIME attachments for binary data
- Getting widely deployed
- Disadvantages:
 - > Fixed cost for small messages
 - > Problem space rather narrow (base64 data)

Fast Infoset (ITU-T/ISO)

- Encode complete XML infosets in binary
- MTOM is just a special case (encoding algorithm)
- Parsing performance: 3-10X
- Compaction performance: 50% on average
- Disadvantages:
 - > Compaction may not be enough for mobile Web

Enabling MTOM

- Using policy assertions in WSDL

```
<wsoma:OptimizedMimeSerialization/>
```

- Using Netbeans 5.5 with support for WSIT

Enabling FI

- Using dynamic content negotiation via the Java system property:

```
com.sun.xml.ws.client.ContentNegotiation  
="none|pessimistic"
```

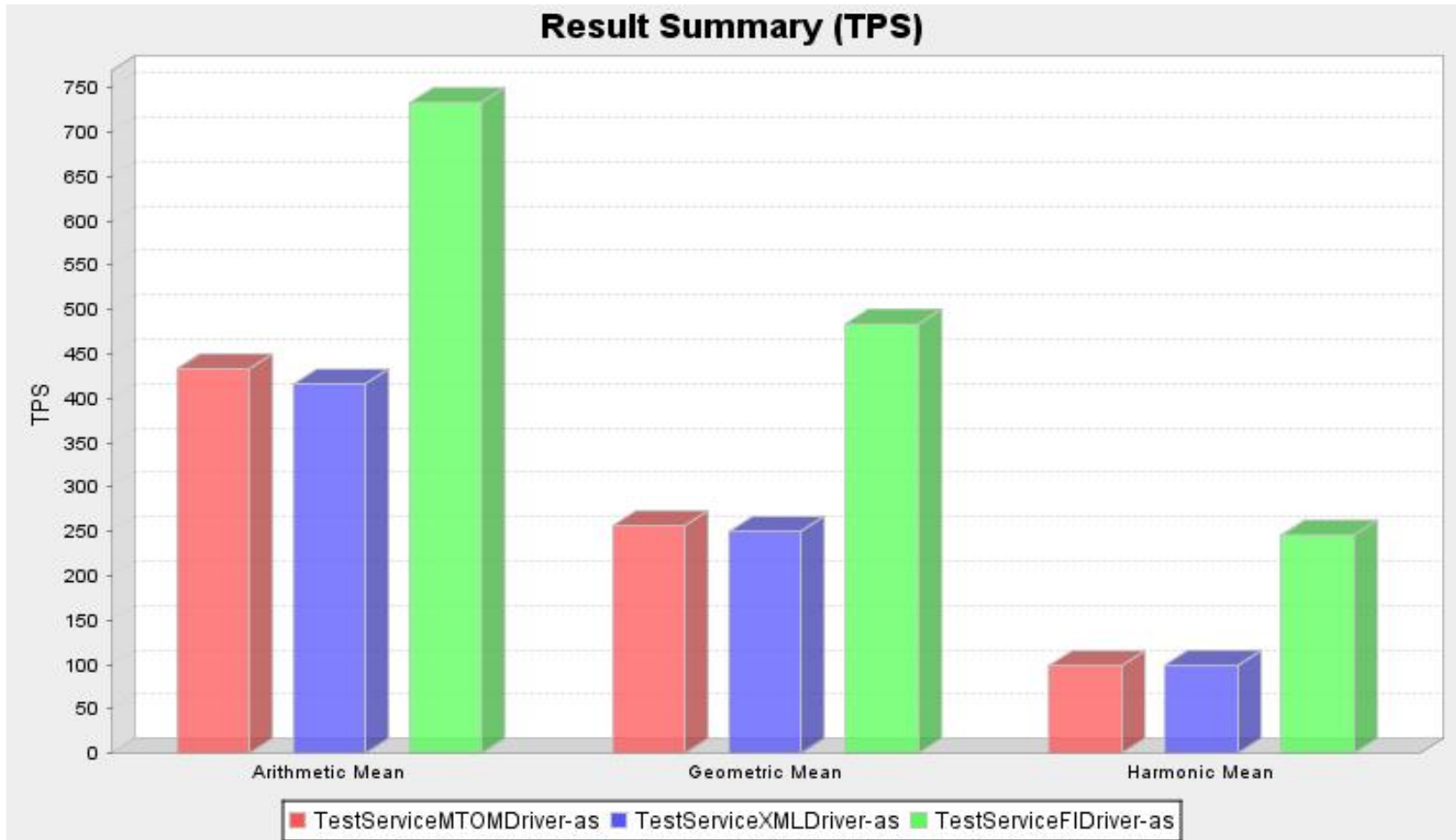
- Using policy assertions in WSDL

```
<fi:OptimizedFastInfosetSerialization  
enabled="true"/>
```

- Using Netbeans 5.5 with support for WSIT

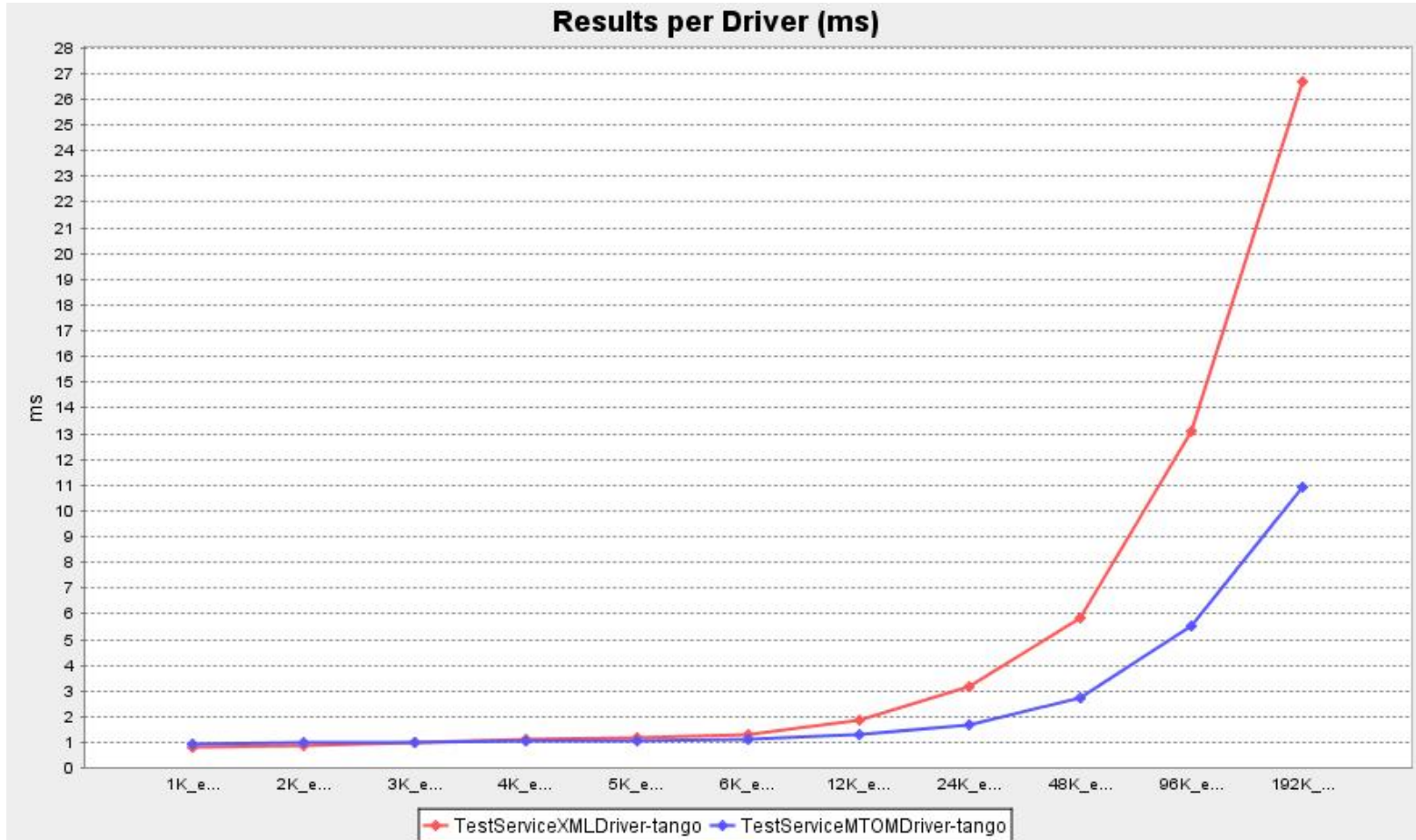
Performance

Over 15 WSDL operations (3 with binary data)



Performance

MTOM vs. XML



Takeaways from this Talk

- How do I address a WS?
- How do I secure my messages?
- How do I know a message has arrived?
- How do I improve encoding performance?
- **How do I ensure a transaction has completed?**

WS-AtomicTransaction

- Based on WS-Coordination framework
- Coordinate activities of **all-or-nothing** type
- Protocols:
 - > Completion
 - > 2 phase commit
- Can leverage the Java EE Java Transaction Service

WS-AT in Action

```
@javax.jws.WebService  
@javax.ejb.Stateless  
@javax.ejb.TransactionManagement(CONTAINER)  
public class Wirerer {  
  
    @javax.jws.WebMethod  
    @javax.ejb.TransactionAttribute(REQUIRED)  
    void wireFunds(...) throws ... {  
        websrvc1.withdrawFromBankX(...);  
        websrvc2.depositIntoBankY(...);  
    }  
}
```

Enabling WS-AT

- Using policy assertions in WSDL

```
<wsat:ATAssertion wsp:Optional="true"/>
```

- Using Java EE JTA annotations

```
@javax.jws.WebMethod  
@javax.ejb.TransactionAttribute(REQUIRED)  
void transferFunds(...) throws ... ;
```

- Using Java EE **user** transactions

Other technologies in WSIT

- Bootstrapping technologies:
 - > WS-MetadataExchange
 - > How do I find a WS description file?
 - > WS-Trust
 - > How do I obtain security certificates?
- State-sharing optimizations:
 - > WS-SecureConversation
 - > How can I optimize multiple message exchanges using WS-Security?

Takeaways from this Talk

- How do I address a WS?
 - > WS-Addressing
- How do I secure my messages?
 - > WS-Security
- How do I know a message has arrived?
 - > WS-ReliableMessaging
- How do I improve encoding performance?
 - > MTOM and FI
- How do I ensure a transaction has completed?
 - > WS-AtomicTransaction

Conclusions

- WSIT technologies are SOA enablers
- Most of these technologies work in the background
 - No new APIs
 - Excellent support in Netbeans
- Best integration with Sun's Application Server
- Proven interoperability with MCF
- Live development @ wsit.dev.java.net
 - Please come and join us!

Interoperability with MCF

September '06 Plugfest Results

- Technologies tested:
 - > WS-Addressing, WS-AtomicTransaction, WS-Security, WS-Trust, WS-SecureConversation, WS-ReliableMessaging, WS-MetadataExchange
- WSIT → MCF and MCF → WSIT
- 188 of 197 tests passing (95%)

References

- Project Glassfish (AS):
 - > <https://glassfish.dev.java.net/>
- WSIT:
 - > <https://wsit.dev.java.net/>
- Fast Infoset:
 - > <https://fi.dev.java.net/>
- Performance
 - > <https://japex.dev.java.net/>
- Blogs:
 - > <http://blogs.sun.com/theaquarium>

WSIT on Netbeans 5.5

Q & A



Web Services Advanced Topics: Reliability, Interoperability, Performance

Santiago Pericas-Geertsen
Sun Microsystems

Santiago.PericasGeertsen@sun.com



**UNLOCK
OPPORTUNITY**

What will you open?

SUN TECH DAYS 2006-2007
A Worldwide Developer Conference