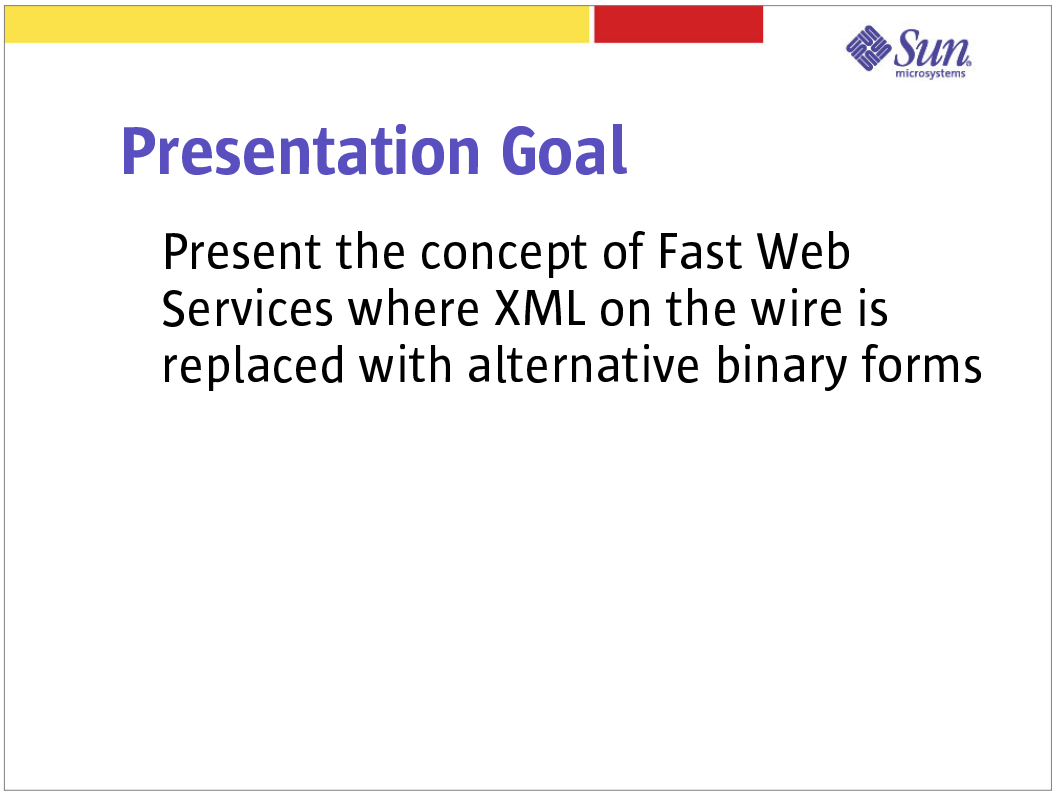
Fast Web Services

Paul Sandoz, Santiago Pericas-Geertsen, Kohsuke Kawaguchi, Marc Hadley

Sun Microsystems



Our work in the area of Binary Interchange of XML Information Sets has been mainly focussed on improving Web Services performance.

XML is rather verbose when compared to existing binary based protocols. XML's verboseness may have advantages in some cases but this may constrain where Web Services may be used.

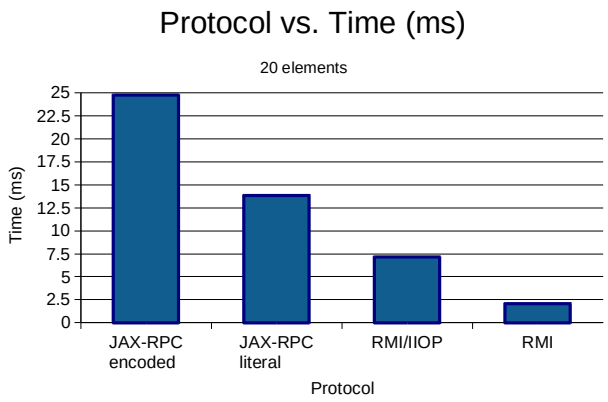
This presentation describes an approach to improving Web Services performance while maintaining the advantages of SOAP, WSDL and associated technologies. Something we're calling Fast Web Services.

Contents

- Current performance data
- Goals and use cases
- Web Service encodings
- Technology and standards
- Java prototype

Current Performance Data

Loopback request/response latency



All performance data was obtained using JDK 1.4.0 and JWSDP 1.0. The -server JVM option was used for client and server.

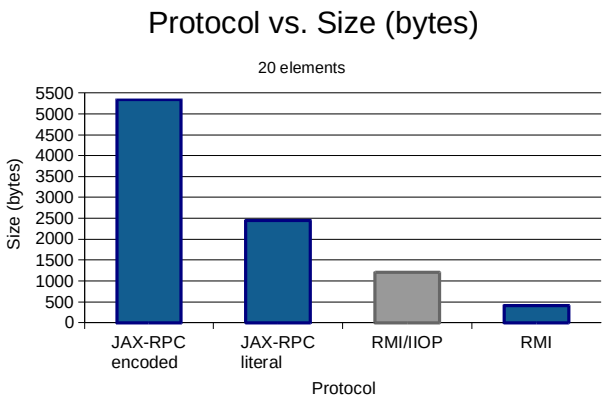
JAX-RPC encoded is the test for SOAP rpc/encoded style/use.
JAX-RPC literal is the test for the SOAP rpc/literal style/use
RMI/IIOP is the test for using standard CORBA facilities in the JDK.
RMI is the test for using the Java Remote Method Protocol (JRMP)

A simple structure of data was chosen such that it could be represented for all four tests. The structure consisted of an array of 'elements', where an 'element' contained a string, integer and boolean with values of 'this is a string', 12345678 and true respectively for all tests. For the results presented the size of the array consisted of 20 elements.

SOAP rpc/encoded is slow, which is recognized by many (see Frank Cohen's PushToTest web site for a good article on the subject). RMI is over five times faster that JAX-RPC literal. Can we get Web Service latency close to RMI? This is our goal.

Current Performance Data

Message size



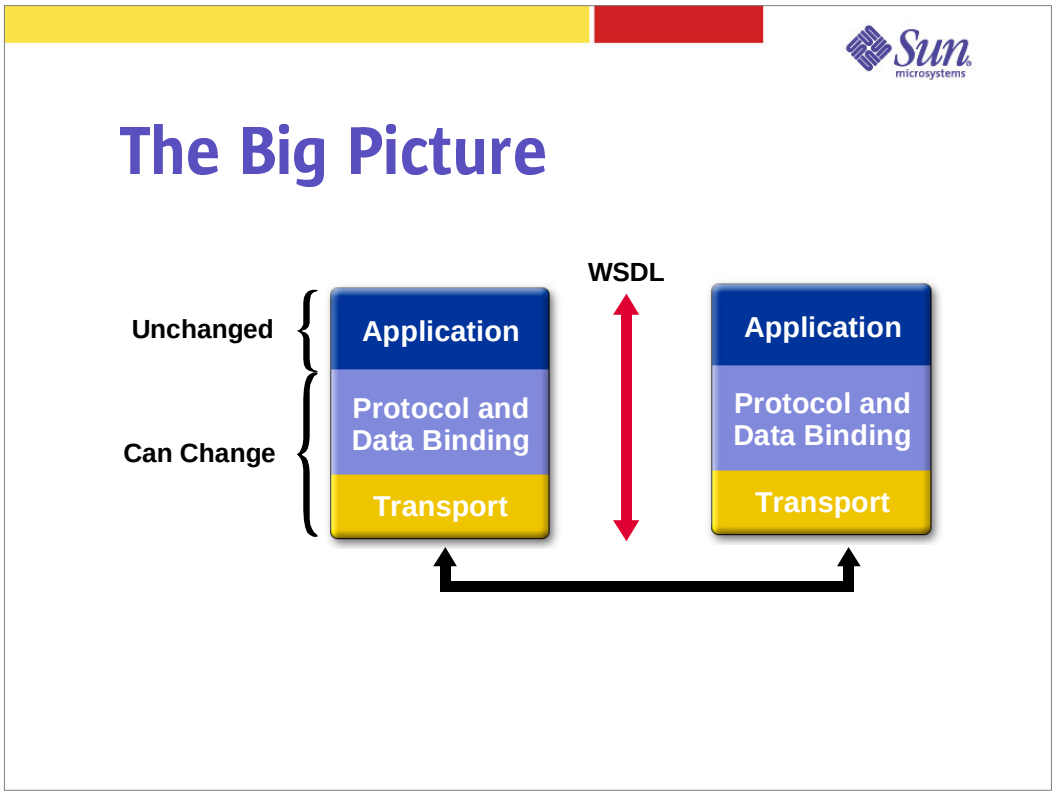
Although not a general rule it appears that the size of the message (not including the protocol meta data, for example HTTP headers) is proportional to the loopback latency time. Based on this assumption, the size of the RMI/IIOP message was estimated (hence the grey) due to difficulties obtaining the actual size.

The inefficiency of SOAP rpc/encoded messages is apparent when compared to the rpc/literal counter-part. SOAP rpc/encoded messages are more verbose due to 'id' and 'xsi:type' attributes.

RMI is approximately 5x smaller than JAX-RPC literal. As for latency can we get Web Service message size close to RMI ?

Cross platform interoperability is one of the key benefits of Web Services. Its essential to use as much existing standardized technology as possible.

The impact on the developer when using Fast Web Services should be minimal.



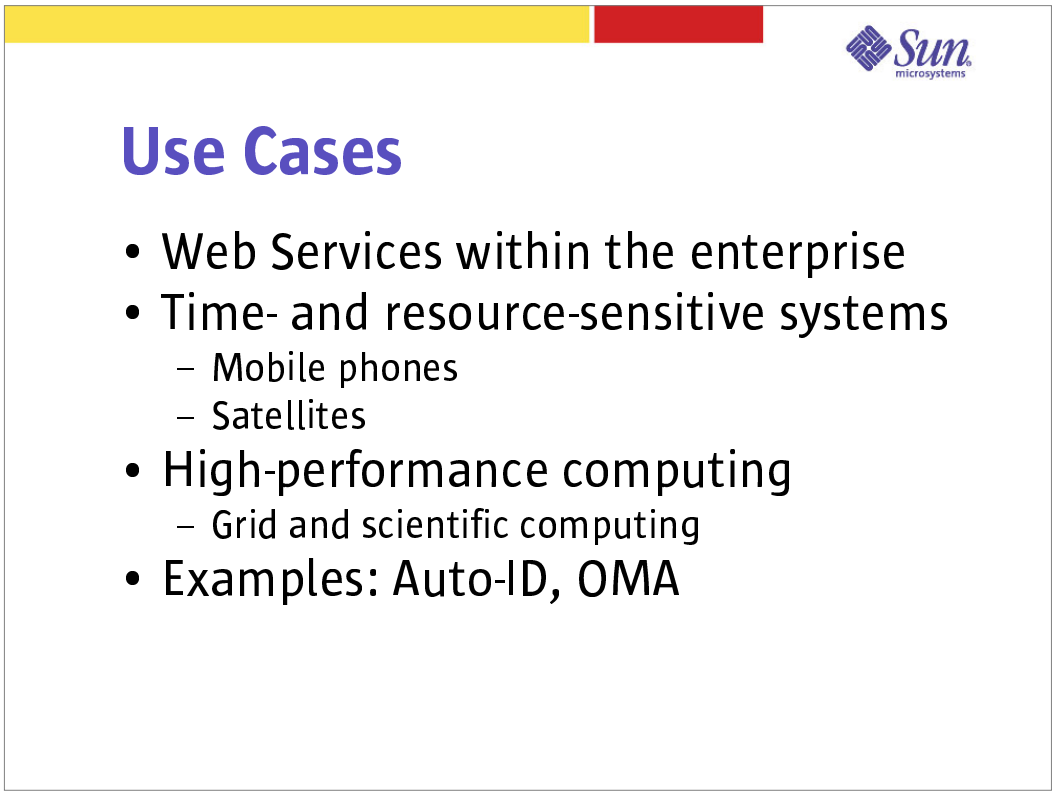
Fast Web Services should not impact the developer. In this respect a client or service should not have to use two different APIs. The implementation stack should hide the details and it is the onus of stack developers to ensure that this is the case (the number of stack developers will be much smaller than the number of Web Service developers). This also means that WSDL should not be required to change in any radical fashion, for example with the addition of new bindings, such that existing WSDL contracts can be used without modification.

A Web Services developer should be able, at the flick of a switch, to state “I want my service to go Faster when talking to Fast peers”

Fast needs to clearly define the interoperability between Fast peers. In addition it needs to define the interoperability with existing deployed Web Services that do not support Fast. The philosophy is: use Fast when possible, use XML otherwise.

Fast Web Services is not Java specific, it is designed to be both platform and programming language independent, just like existing Web Services and CORBA.

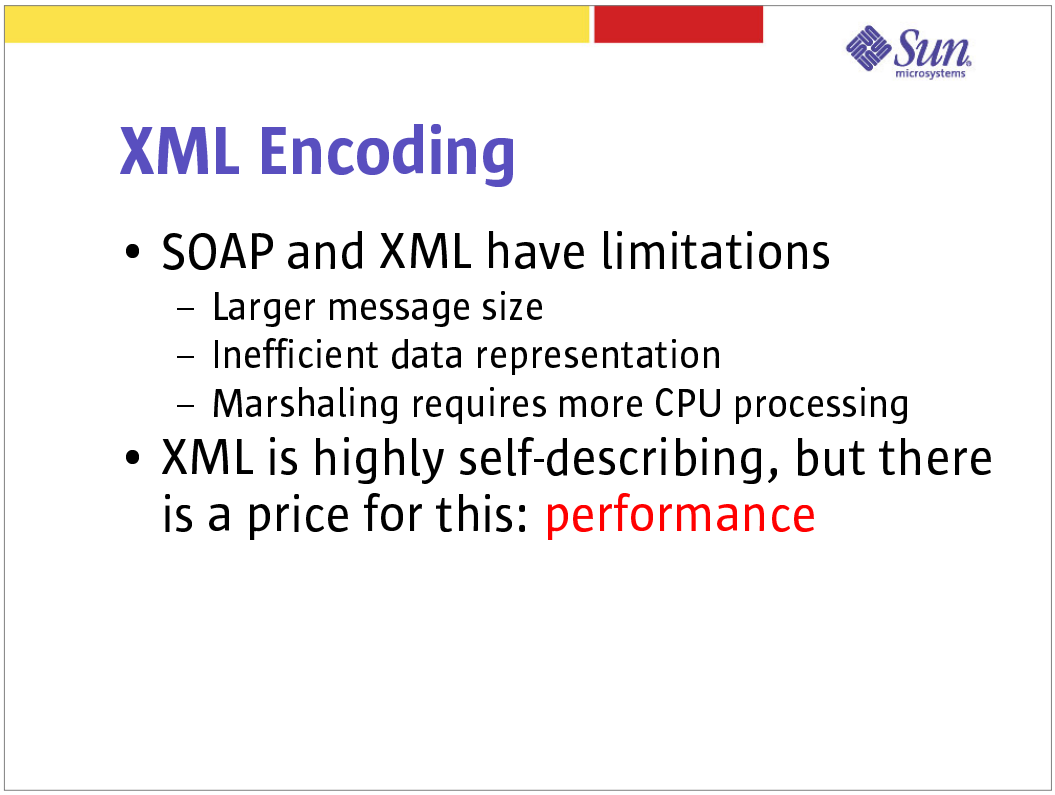
End-to-end support is also considered important. Its important that Fast can be supported in constrained environments such as mobile phones using J2ME Web Services (JSR-172). Thus a gateway does not need to transform between formats.



Web services within the enterprise, e.g. for use in Enterprise Application Integration, dominate today. The reasons for this: ease of deployment, use of stable specifications and single security domain are also attractive to Fast.

Web Services for satellites where real-time communication is required provides for an interesting domain. The tools and technologies of XML-based messaging were recently evaluated for a large European satellite constellation project. The technical support team ruled out XML messaging and its tools because these were not suitable for efficient transfer of data in real time. Instead they chose to concentrate on tools and technologies that are based on binary messaging (using technology that underlies Fast).

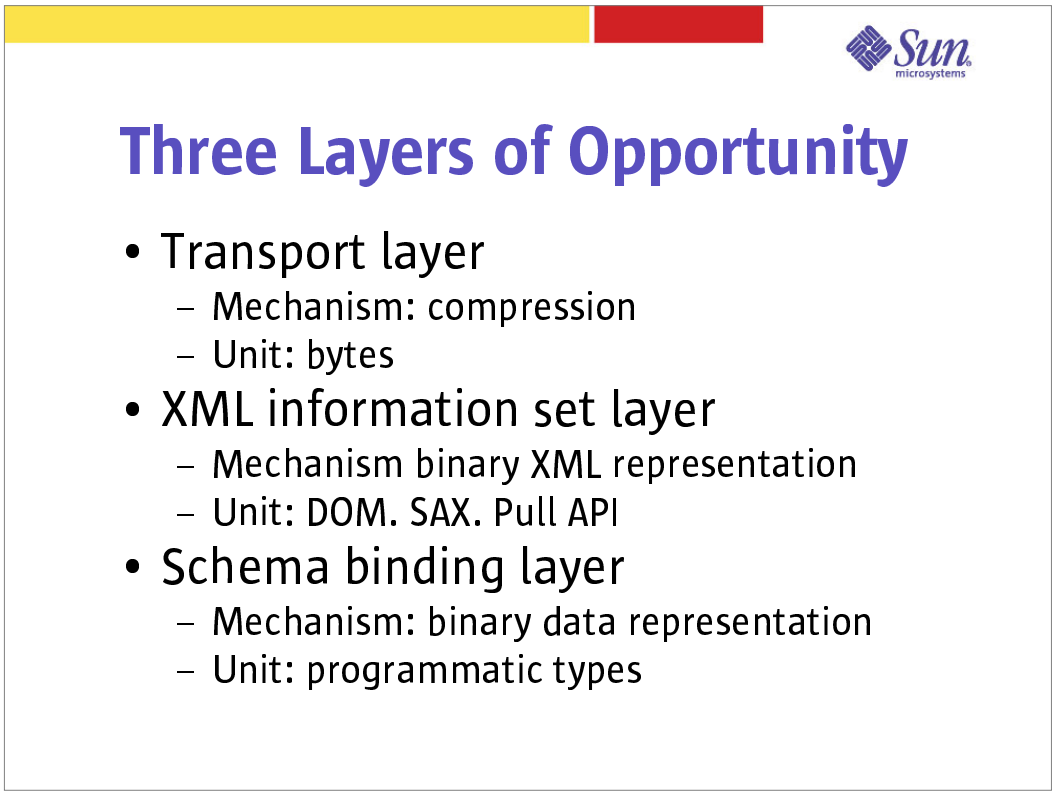
Auto-ID is an initiative to standardize the processing and management of Radio Frequency ID, RF-ID, systems. The Open Mobile Alliance are defining how non-intermediated Web Services can be used for value-add services. Both these domains will involve small devices communicating to servers and the potential scale of the systems could be very large.



As seen from the results showing message size, SOAP messages (both encoded and literal) are larger than the binary equivalent.

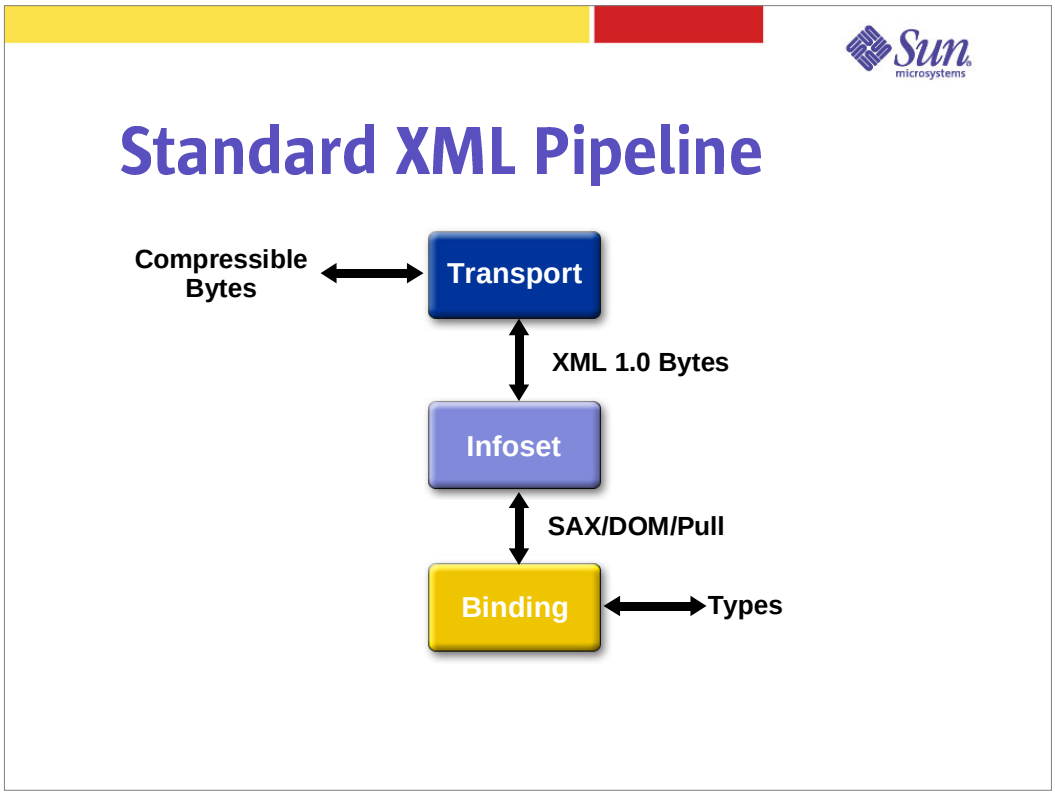
Data such as numbers, integer and real, are inefficiently represented. This could be an issue for scientific computing where arrays of real numbers need to be encoded on the wire.

XML is verbose. There are many advantages to XML as is apparent by its astonishing success, but performance can be an issue.



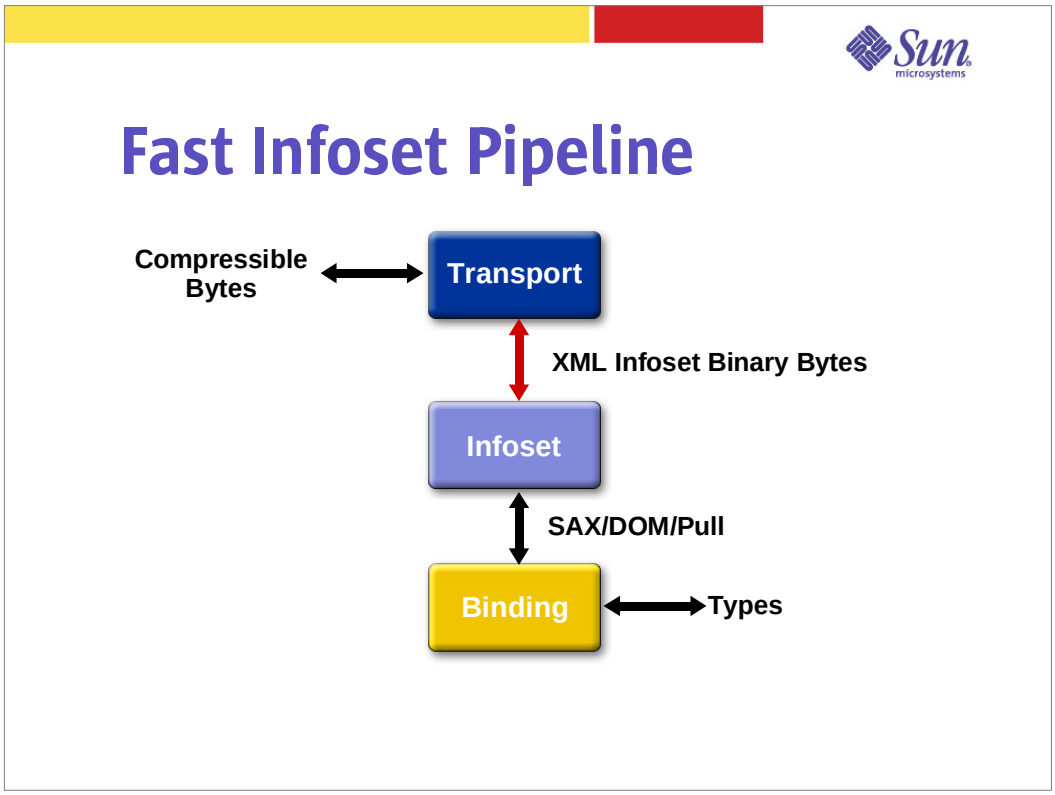
A 'stack' processing XML received over a network essentially has three layers to it: the transport layer, such as HTTP; the XML information set layer, where an XML API is used to access the information set; and a schema binding layer where XML information items may be bound to programmatic types.

Each layer has a mechanism that may be used to improve the performance of XML processing and a unit of 'data' that it operates on.



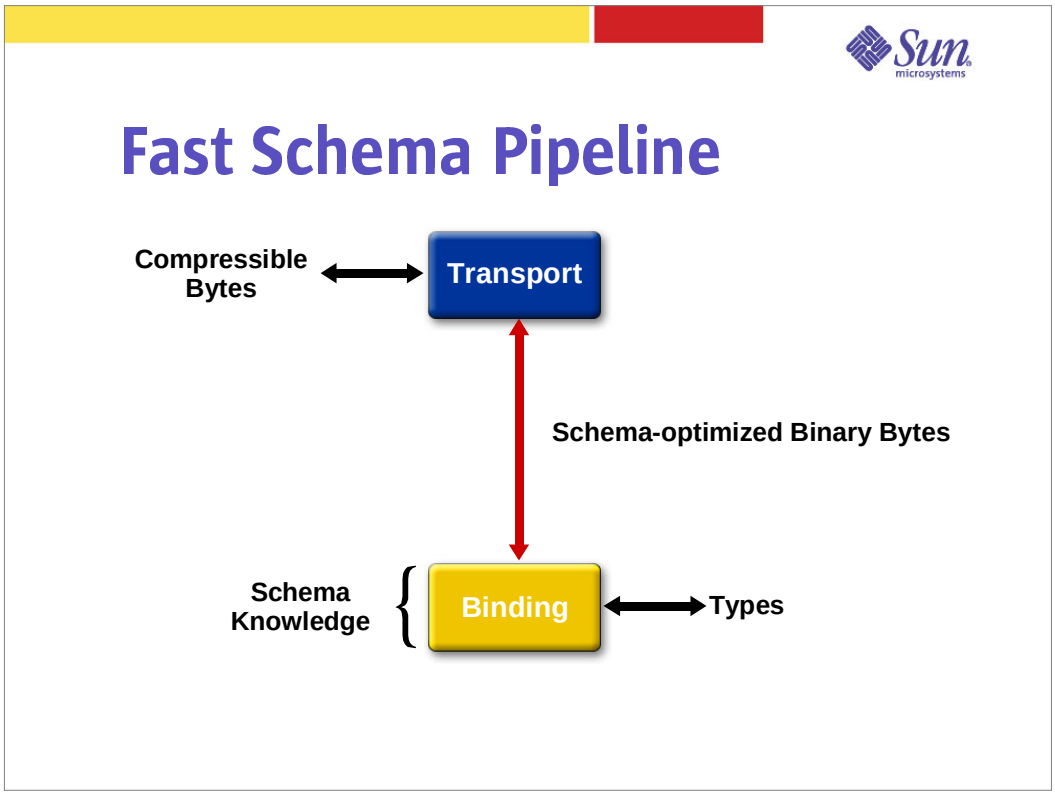
A standard pipeline through the stack may take compressed bytes via a transport, decompress them to XML 1.0 set of bytes which are then passed to an XML parser where the XML information items are presented using an XML API to the binding layer, which converts the XML to types.

Applying compression may have mixed results. Compression will generally result in more work done by a sender and receiver if the rate of transfer of information is greater than the time to decompress i.e. If the bandwidth is there compression may not help. Compression may help when using low bandwidth links. However, there is still the cost of compressing.



A Fast Infoset pipeline replaces XML 1.0 with a self-describing binary representation of the XML information set where no information is lost.

This may help matters, however the binding layer still needs to do work. If the transport to Infoset layer processing is small compared to the Infoset to binding layer then this may only give a moderate increase in processing performance. This has been observed this to be the case for SOAP/RPC usage.



The Fast schema pipeline skips the XML information set layer such that the data from the transport layer can be passed directly to the binding layer.

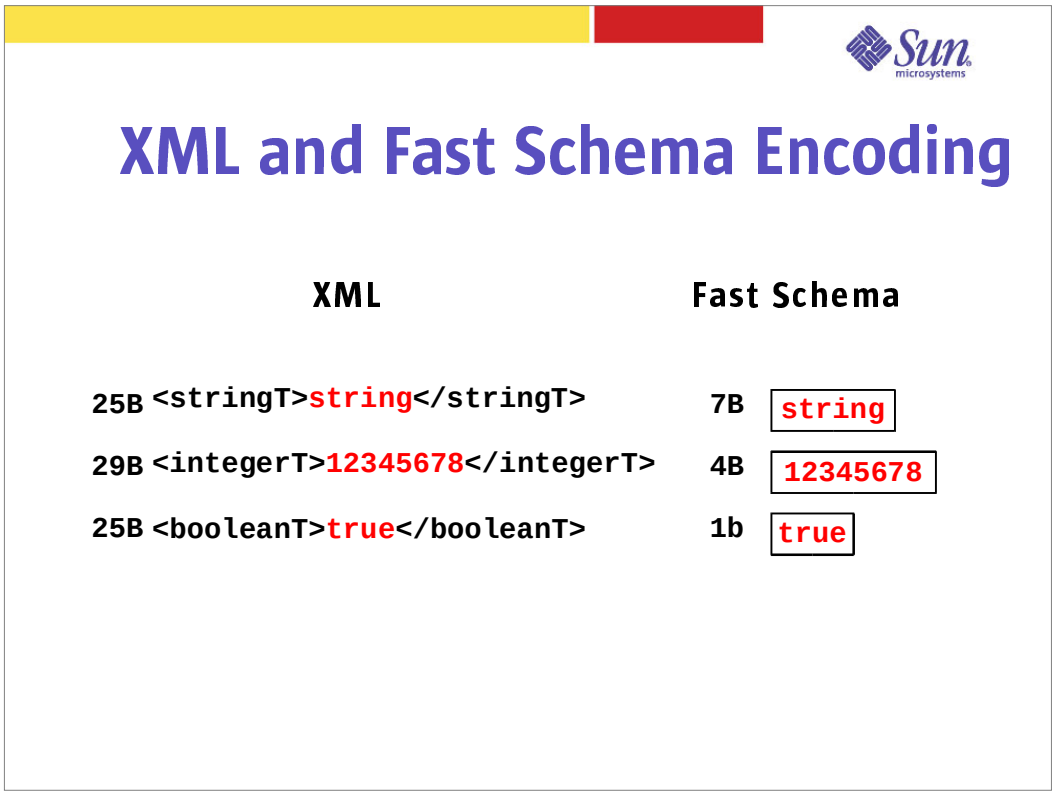
This represents the most efficient and performant route. The data is encoded in a form that is efficient for the binding layer to process. To encode or decode requires knowledge of the schema. Note that such knowledge is required by the binding layer regardless of whether the Fast schema encoding or XML encoding is used.

Note that the Fast schema encoded data may also be compressed. If the size of the encoded data is less than the XML equivalent then compression will take less time and may improve on the resulting compressed size of the XML.

XML compressors such as XMill can improve on the compression size of standard compression algorithms applied to XML data. A study has shown that compressing Fast schema data results in similar sizes to that of XMill.

Example: Schema Fragment

```
<complexType name="structType">
    <sequence>
      <element name="stringT" type="xsd:string"/>
      <element name="integerT" type="xsd:integer"/>
      <element name="booleanT" type="xsd:boolean"/>
    </sequence>
</complexType>
```



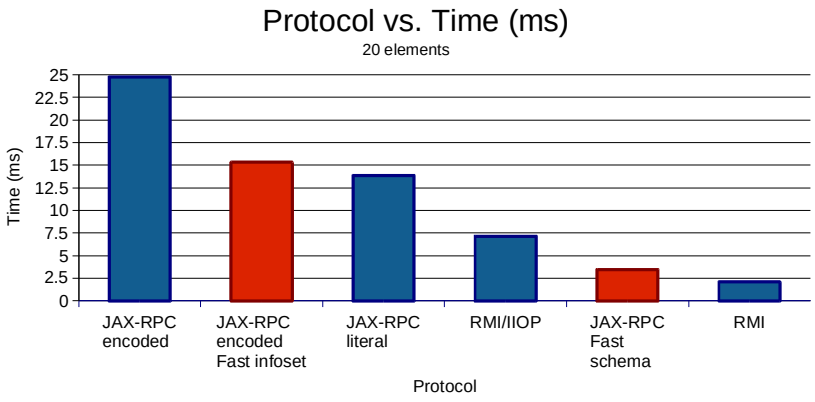
Instance of XML and fast schema encoding, given the schema fragment in the previous slide, are shown.

The data is highlighted in red. On the left are the size of the respective chunk of data in 'B'ytes or 'b'its.

It is difficult to show how binary data is represented visually and meaningfully. The string value is UTF-8 encoded with a length prefix. The integer value is length prefixed and encoded as 4 bytes. The boolean value is encoded as only 1 bit since the length is known.

Fast Performance Data

Loopback request/response latency



Highlighted in red are additional results to those shown previously where Fast technology has been applied.

The JAX-RPC encoded Fast infoset test uses an alternative binary representation of the infoset, for the same JAX-RPC encoded test. The underlying technology used in this case was a modified version of Denis Sosnoski's XMLS., which was integrated into Sun's JAX-RPCs pull parser implementation.

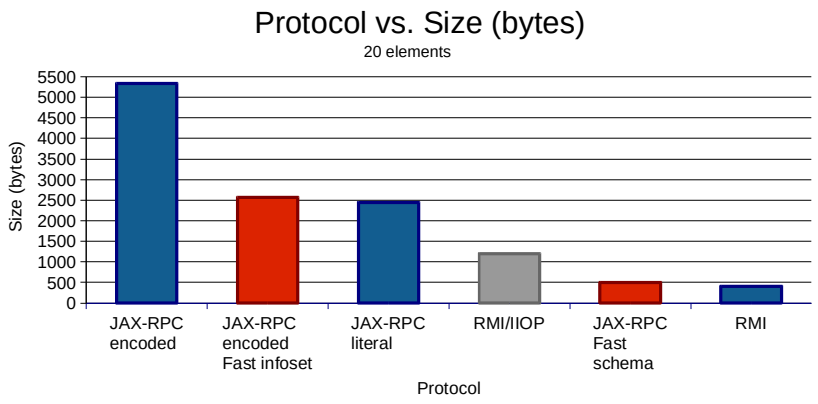
The JAX-RPC Fast schema test uses the non-self describing encoding, described briefly in the previous slide, for the same JAX-RPC literal test. The same HTTP client transport and Tomcat servlet engine is reused.

The Fast infoset mechanim improves on the XML for JAX-RPC encoded by 40%.

The Fast schema mechanism is 4x faster than the XML for JAX-RPC literal. Given that JAX-RPC Fast schema test is using HTTP and is platform independent Fast is performing well when compared to RMI.

Fast Performance Data

Message size



It can be observed that the Fast infoset size is approximately half that of the SOAP rpc/encoded XML equivalent.

The size of the Fast schema encoding is similar to RMI.

The apparent rule of 'message processing is proportional to message size' seems to be holding for these results.

Another test needs to be performed on applying Fast infoset to the JAX-RPC literal test. Given the existing results it is believed that a similar 40% reduction may be obtained.

Encoding Comparison Matrix

	XML	Fast Infoset	Fast Schema
Size	Large	Medium	Small
Processing	High	Medium	Low
Self-describing	Yes	Yes	No
XML API support	Yes	Yes	Yes

Fast Schema Vs. Fast Infoset

- Fast schema good for binding applications
  - XML infoset layer is hidden
  - Schema are known
- Fast infoset good for generic XML processing
  - Preserves XML data
  - xsd:any content

Fast schema cannot be used when there is xsd:any content since the schema is not known. In this case the Fast infoset can be used.

WS Encoding: Fast Schema

- Require schema to process message parts
  - WSDL is used by client and service
- Require Fast SOAP
  - Binary SOAP representation
  - Semantics & processing model preserved
- Fast schema Wss are efficient, but there is a price for this: loss of self-description

Technology Requirements

- Consistent non-specific encoding technology
  - Fast infoset, Fast schema, and Fast SOAP
  - Not specific to application
- Proven use in network communications
  - Large scale deployment
- Platform and programming language independent
- Existing standards
  - Royalty-free and open

Note that the modified XMLS used for the Fast infoset tests is not consistent with the encoding technology used for Fast schema.

Abstract Syntax Notation One (ASN.1)

- Schema language for abstract type system
- Multiple encoding rules
  - Types are independent of encoding
- Royalty-free standards, at ITU-T/ISO
- In development for nearly 20 years
- Implementations in Java, C, C++
- Extensively used in telecom industry

ASN.1 may seem obscure to Web services specialists, but it is used extensively in the telecommunications industry. For example, ASN.1 is used in mobile phone networks to help transfer control between network cells. It also plays a critical role in the central nervous system of the telephone network when routing data is modified.

Fast Encoding and ASN.1

- Fast infoset encoding
  - ASN.1 Schema for the XML infoset
- Fast schema encoding
  - W3C XML Schema to ASN.1 mapping
- Fast SOAP encoding
  - ASN.1 Schema for SOAP
- Packed Encoding Rules (PER)
  - Most compact and CPU efficient
  - Other rules could be used (e.g., DER)

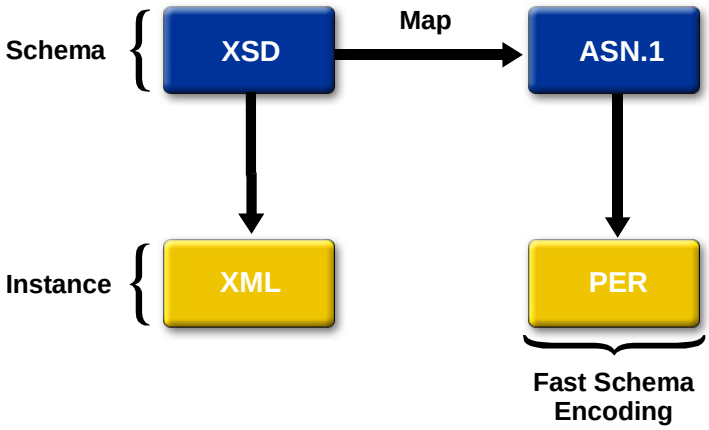
ASN.1 Standards

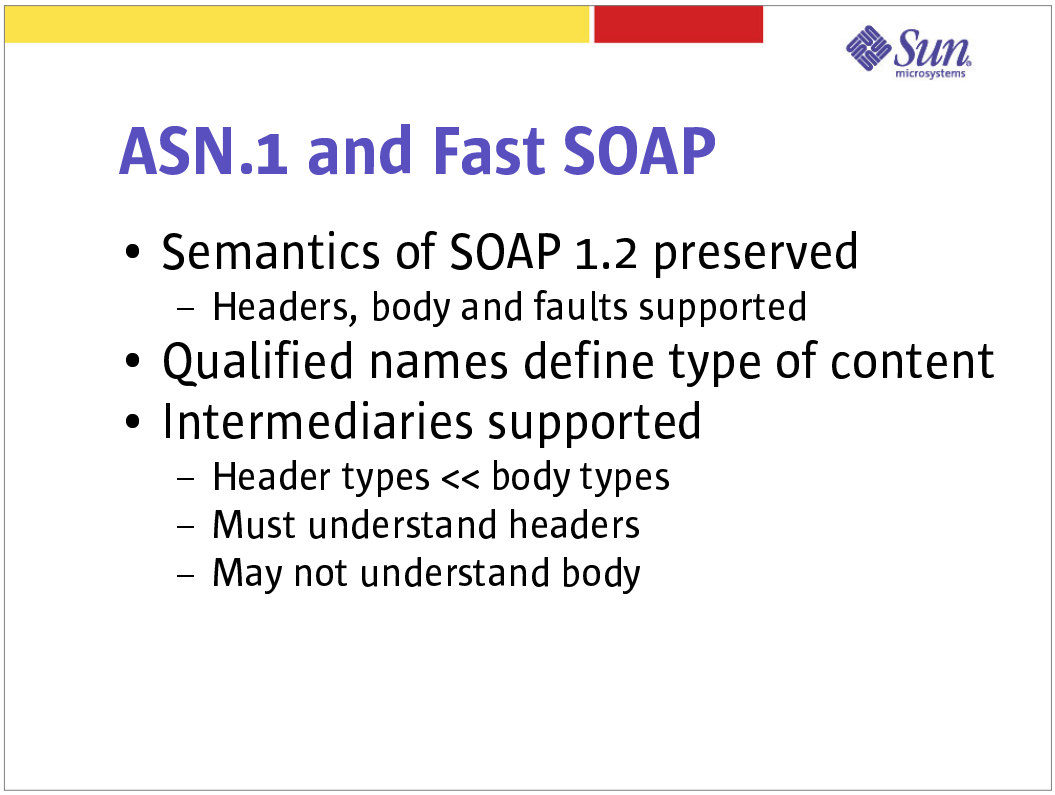
- Existing and progressing
  - PER: X.691
  - W3C XML Schema to ASN.1 mapping: X.694
- X.695: proposed standards activity at ITU-T/ISO
  - "ASN.1 Support for SOAP, Web Services and the XML Information Set"
  - Sun is actively participating
  - ASN.1 Schema for SOAP
  - ASN.1 Schema for the XML infoset

ITU-T = International Telecoms Union Telecommunication Standardization sector

The ASN.1 work is performed under Study Group 17 (SG-17), which is charged with progressing ASN.1 until the end of 2004.

XSD and ASN.1, XML and PER





Fast SOAP defines an ASN.1 schema for SOAP.

Just as a qualified name defines the contents of a SOAP body, header or fault a qualified name is used to define the encoded content in the ASN.1 schema.

SOAP headers and SOAP intermediaries can be supported. An intermediary that must understand headers has to be able to bind (either formally using a tool or informally, for example, hand coded) to the header content such that it can be processed meaningfully. Thus intermediaries will need to know the schema for the header content they must understand.

Security and Fast SOAP

- WS-Security is a problem
  - Sign/encrypt of sub-content using Xpath
  - Performance issue recognized by TC
- Can sign complete header, body or fault content

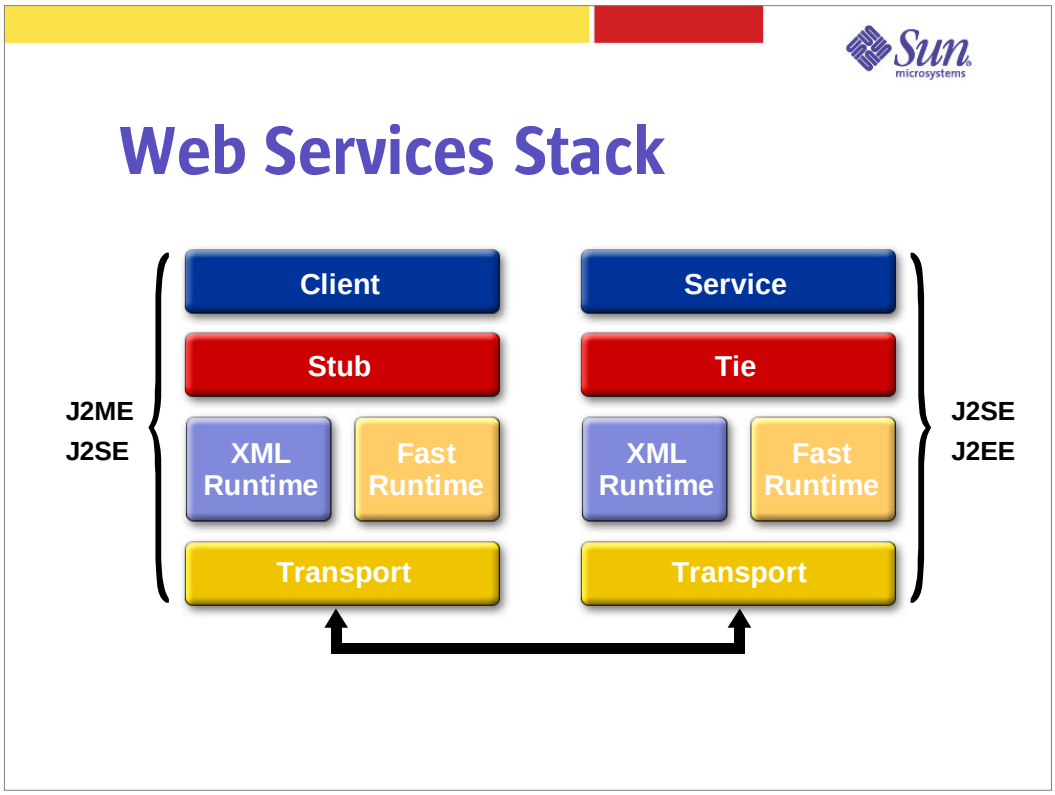
XML DSig provides the capability to sign a subset of an XML document. Subsets are defined using transforms of the source data, E.g. a subset can be defined using an XPath. To perform the transformation requires instantiation of an infoset (or rather an instance of the XPath data model) against which the transform can be applied – streaming cannot always be supported due to the nature of XPath. Such instantiation and transformation can be quite expensive and the WSS TC at OASIS is currently working on a profile of WS-Security that eleminates some of the flexibility inherent in XML DSig to improve performance. Fast has problems supporting the full generality of XML DSig since an inforset is not available. However, it is possible to support a restricted subset of XML DSig (similar to that mentioned above) where the data to be signed consists of some combination of the SOAP headers and body.

Fast SOAP Binding

- Fast SOAP is a different protocol to SOAP
- Dynamic negotiation
  - No modification to WSDL
  - `soap:binding` may be reused
- Explicit declaration
  - Annotate `soap:binding` with `wsfast:binding`
- One port, two bindings, two representations

The syntax of the WSDL binding can be reused for the Fast SOAP binding and no changes to WSDL are required. This means that it is possible for one port type bound to the soap binding to support Fast and XML. This is important since it does not require the developer to have two ports for the same port type bound to two different bindings.

A client and a service could potentially negotiate to communicate using Fast or XML based on the content described by multimedia MIME types. The alternative is to allow for the service to annotate the soap binding with Fast information, thereby the client and service know what they can send and receive.



This depicts the ideal representation of a Web Service stack that supports both XML and Fast.

Stubs and ties are independent of the Fast or XML runtime, and will mediate with a runtime based on pre-generated encoders and decoders (not shown, and in this case JAXB types).

This enables the client to dynamically switch runtimes and for the service to process concurrent requests from both runtimes.

Fast and XML Interoperability

- Clients and services can know capabilities
- Intermediaries use content negotiation of transport
  - E.g., HTTP content negotiation
  - Optimistic
  - Pessimistic
- Transcoding intermediaries

The concept of a transcoding intermediary is mooted, based on using forms of content negotiation.

Schema Drift

- Client and service may use slightly different schema
  - No versioning
  - Addition or removal of elements/attributes
- XML can support certain changes
  - Not a panacea
- Fast schema cannot support schema drift
- Solution: Fingerprint data structure and types

Schema drift is an interesting problem and the versioning of Web Services and the schema used is a difficult problem. It is recognized that developers may not correctly apply a versioning strategy, which results in clients and services becoming out of sync with respect to the schemas they use. The use of XML can enable the support of certain changes due to the self-describing information. However, this is not a panacea and will effect the performance of a 'lax' processor capable of dealing with extra or missing information.

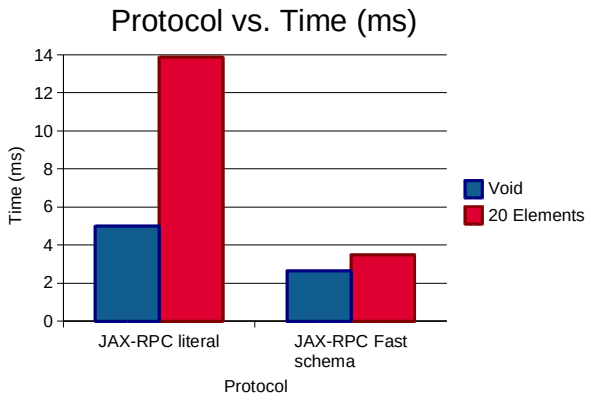
Fast cannot support schema drift since there is no self-descibing information, and the same applies to CORBA, RMI and DCOM. The solution is to generate a fingerprint of the schema used to describe messages. In essence this is an MD5 hash of a canonical representation of the schema. A client or service can send the fingerprint along with the content and if two finger prints do not match the client can revert to XML.

Java Prototype

- Implementation of Fast WS
- JAXB marshaling of WSDL message parts
  - Contains X.694 mapping logic
- J2ME and JAX-RPC clients
- Doc/lit and RPC/lit
  - Static stub/tie generation
- Headers not supported yet

Performance Results

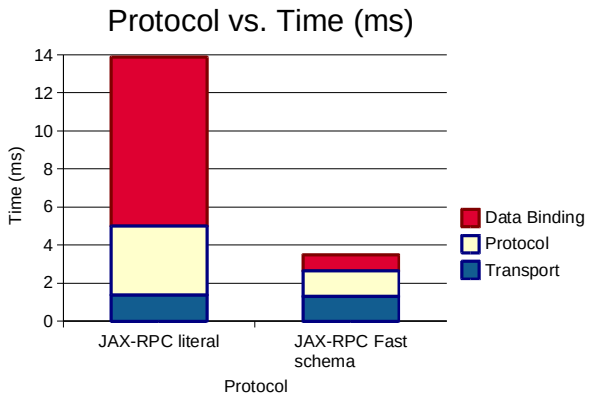
Loopback request/response latency



This slide presents the JAX-RPC literal and JAX-RPC Fast schema tests in more detail. In addition to the 20 element test the void test is also presented, which represents a SOAP message with no body content.

Performance Results

Time spent in layers



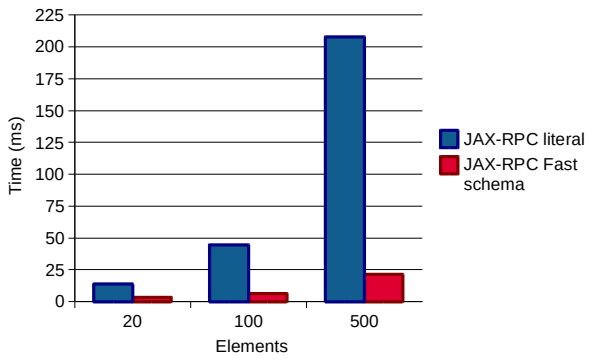
Given the void tests and obtaining request/response times for basic HTTP POST, of data with equivalent sizes to the SOAP envelopes, it is possible to present an approximate breakdown of where time is being spent in the various areas of transport, protocol and data binding.

The transport represents the HTTP transport and HTTP processing of the client and the Tomcat servlet engine. The protocol represents where the SOAP envelope is processed. The data binding layer represents where the SOAP body content is encoded and decoded from and to Java types respectively.

Fast improves performance best in the data binding layer, where it is almost 10x faster.

Performance Results

Increasing size of message



Assuming that the transport and protocol are near fixed costs it should be possible for the times to tend towards that of binding when the data being bound is large.

It can be observer that Fast gets faster as the number of elements is increased from 20 to 100 to 500, which gives 4x, 7x to 9.5x increase respectively.

Observations

- Data binding layer benefits most
- Protocol also benefits
- Transport is almost a fixed cost
  - Further investigation required
- Gets faster with larger content
  - Nearly order of magnitude for 500 elements

Summary

- Two alternative 'optimizations' on the wire
  - Fast infoset
    - 1.6x faster
  - Fast schema
    - 4-10x faster
- Participating in ITU-T/ISO for Fast WS

Paul Sandoz, Santiago Pericas-Geertsen, Kohsuke Kawaguchi, Marc Hadley