The gSOAP and Grid Service Architecture for Distributed System Integration

Amarjeet Singh, Balvinder Singh, Sanjeev Thakur,
1Assistant Professor, Department of Computer Science, SIMT, Kashipur, Uttarakhand, INDIA.
2Assistant Professor, Department of Computer Science, SIMT, Kashipur, Uttarakhand, INDIA.
3Professor, Amity School of Computer Science (ASCS), Sector 44, Noida, U.P, INDIA.

ABSTRACT
When designing distributed web services, there are three properties that are commonly desired: consistency, availability, and partition tolerance. It is impossible to achieve all three. This paper studies the usability, interoperability, and performance issues of SOAP/XML-based Web and Grid Services for scientific computing. Several key issues are addressed that are important for the deployment of high performance and mission-critical SOAP/XML-based services.

I. INTRODUCTION
Web services are expected to be highly available. Every request should succeed and receive a response. There are many real world problems faced when a service goes down for example when an E-Trade web site goes down then many potential legal difficulties may be faced. This problem is exacerbated by the fact that when a web-site is most needed then it is likely to be unavailable. Today the goal of most web services is to be as available as the network on which they run: if any service on the network is available, then the web service should be accessible. By supporting the creation, deployment, and dynamic discovery of distributed applications, today the Web Services framework is gaining momentum as an approach to distributed computing. The popularity of Web Services can be contributed to the certain transition from people to software applications to access public, commercial, and government services on the Internet. This creates new opportunities for Grid applications. Another important feature of the framework is the high level of interoperability and service compositionality across platforms and programming languages. The framework also provides a practical infrastructure for Internet computing that includes protocols for enhanced routing capabilities such as multi-hop routing with intermediaries and the streaming of Direct Internet Message Encapsulation (DIME) attachments.

Also available are Web Service security extensions such as HTTPS, WS-Security, SOAP digital signature, and GSI. It is widely professed that Web and Grid Services can suffer severe performance penalties by adopting the XML based SOAP protocol for transporting large volumes of data. Several papers have reported on the performance of scientific computing with Web Services, such as a performance investigation of SOAP for scientific applications that utilize large floating point arrays, a comparison of the latency of SOAP Web Services implementations, an evaluation of Web Services based implementations for Grid RPC, and the design and implementation of the gSOAP toolkit for efficient embraced Web Services as a key technology supporting the shift to service-oriented Grid applications. A Grid Service is envisioned as “a stateful service instance supporting reliable and secure invocation, lifetime management, notification, policy management, credential management, and virtualization”. The Open Grid Services Architecture (OGSA) leverages Web Services protocols and additional interfaces to manage Grid service lifetime, policies and credentials, and to provide support for notification as mandated by the OGSA specification.

The usability, interoperability, and performance aspects of Web and Grid Services for scientific computing are addressed in this paper. Usability is largely motivated by the fact that the Web Services framework provides universal access to numerous public and commercial services on the Internet. This creates new opportunities for Grid applications. Another important feature of the framework is the high level of interoperability and service compositionality across platforms and programming languages. The framework also provides a practical infrastructure for Internet computing that includes protocols for enhanced routing capabilities such as multi-hop routing with intermediaries and the streaming of Direct Internet Message Encapsulation (DIME) attachments.
SOAP/XML Web Services. However, these studies did not address the key issues that can affect performance the most. The key issues that may prevent a successful deployment are the overhead of XML encodings with SOAP/XML and the message passing latencies of HTTP over TCP/IP. These issues are discussed here by exploiting XML schema extensibility for defining optimized XML representations for numerical data and by message passing optimizations such as chunking, compression, routing, and streaming media techniques.

There are many questions that help us to identify several key issues for deploying high performance and mission-critical SOAP/XML-based services like, How does SOAP/XML as a generic data representation compare to more specific XML representations for scientific data such as MathML and XSL, for example? What is the performance impact of SOAP/XML Web Services compared to C/C++ applications on the Web? Which alternative XML representations for numerical data are available and what is the impact on communication latency and performance?

**II. WEB AND GRID SERVICE TOOLKITS**

Now a day’s more than seventy SOAP Web Services toolkits are available for a variety of programming languages and platforms. Because production-quality scientific applications are written in Fortran1, C, or C++, we will only mention the C/C++ SOAP toolkits. These SOAP Web Services toolkits and libraries for C++ are Borland builder 6 with Delphi, easySOAP, eSOAP, gSOAP, the .NET framework, WASP for C++, and SQLData. The gSOAP toolkit is a platform-independent development environment for C and C++ Web Services [18]. The important design considerations were ease of use and performance in the development of the toolkit. The toolkit provides an easy-to-use RPC compiler that produces the stub and skeleton routines to integrate (existing) C and C++ applications into SOAP/XML Web Services. A unique aspect of the toolkit is that it automatically maps native C/C++ application data types to semantically equivalent XML types and vice versa. This enables direct SOAP/XML messaging with C/C++ applications on the Web. The run-time mapping to XML with gSOAP’s schema-optimized XML parsing techniques has very low overheads and memory usage, which makes gSOAP attractive in high-performance environments and embedded systems. The Globus Toolkit v3 includes the gSOAP toolkit.

gSOAP is used to implement C/C++ bindings for the OGSA-compliant Grid Services. The gSOAP toolkit is also used in the GridLab project [9] with the Globus Toolkit v2 and the GSI plug-in for gSOAP [3] and in the Harness project for distributed computing. Many companies have shown an interest in gSOAP. For example, the gSOAP toolkit is integrated in the IBM alphaWorks Web Services Tool Kit for Mobile Devices.

**III. LOGICAL COMPUTING WITH SOAP/XML**

SOAP is a protocol for remote procedure calling and messaging with XML-encoded application data. But SOAP does not require the use of XML per se. In fact, SOAP supports binary data attachments and remotely referenced data such as objects provided by third parties that are produced or consumed at separate hosts, for example. To create data portals on the Web, these features can be used. SOAP also specifies various usage scenarios, such as one-way message passing, single and multiple request-response invocations, and routing. Consider for example the data transfer illustrated in Figure 1. Data is efficiently transferred from a client to a server with gSOAP’s streaming SOAP/XML and DIME. This form of routing streams the (binary) data encapsulated in DIME from a data source at the client side into a data repository at the server side. This data can be retrieved and stored on disk or processed dynamically by tasks running on separate machines, for example. Note that SOAP/XML with stream-ing DIME file transfers are more flexible than current file transfer protocols such as FTP and GridFTP, because the SOAP/XML messages can be utilized to carry meta-data with the file transfer requests and responses. In addition, Web Services extensions, such as routing and security for example, are orthogonal which means that file transfers can be routed and protected.

SOAP and WSDL are protocols that utilize the richness of the XML schema specification standard for defining XML content. XML schemas define the structure of XML documents by defining document layout and content constraints. The schema notation is agreeable to meta-level data translation and interpretation. In fact, SOAP toolkits already translate WSDL service descriptions into service objects and client proxies by inspecting the WSDL schema type’s parts to determine the parameter marshaling requirements. The SOAP RPC and (to a limited extend) the SOAP literal encoding styles provide a rich set of data types that can be utilized for XML encoding, including primitive types, enumerations, bitmasks, arrays, lists, and records. Using XML schema extension and restriction one can add arbitrary data types with application-specific semantics. Schema extension and restriction introduces an object-oriented approach to data modeling with XML.
Today many XML protocols for scientific data are based on DTDs rather than XML schemas. They lack XML namespace and extensibility properties. The clarity of these protocols is limited to a set of pre-defined data types offered by the protocol. For example, XSIL (Extensible Scientific Interchange Language) [25] and XDMF (Extensible Data Model and Format) [24] both provide encodings for primitive and aggregate types in XML. XSIL concentrates on data structures with a Java-dependent XML format. The XSIL, XDMF, MathML, and BioML XML representations can be easily blended into SOAP/XML. However, the application-specific XSIL and XDMF representations have a limited set of data types and offer no means to utilize schema extension to define optimized and application-specific XML layouts.

The XML schema standard provides a basis for programming language bindings for XML as demonstrated by gSOAP and the .NET framework, for example. At compile time gSOAP generates XML schemas and at run time automatically performs data mappings. The toolkit guarantees the preservation of the logical structure of graph-based data which is difficult to achieve with hand-written conversion routines (often referred to as wrappers). Scientific applications typically deal with large symbolic or numerical data sets. Symbolic data can be of two types; one is text-based and second is graph based, such as symbolic expressions. Using SOAP RPC encoding of nodes, graph-based data can be represented in XML such as “structs” and co-referenced nodes as “multi-referenced objects”. The use of XML graph nodes to representing symbolic expressions in XML can be based on existing XML standards, such as MathML and CML (Chemical Markup Language).

In many different forms of XML, flat data structures such as numerical vectors and matrices can be represented including sparse SOAP arrays, UTF8-encoded strings, and hexadecimal and base64 binary encodings. These structures don’t necessarily require a generic SOAP array representation with values represented in decimal notation. The loss of floating point precision and the encoding overhead of floating point numbers in decimal often prohibits the exchange of numerical data in plain text form.

IV. LATENCY AND PRESENTATION

The choice of XML encoding for the data of a SOAP/XML Web Service application can have a significant impact on the communication latency and overall performance of the service. The XML representation of floating point arrays, for example, can be as large as ten times the original binary representation. But, the total message length is not always the determining factor of performance but the time to convert data in XML is, because it introduces a significant overhead. This overhead is also evident in the linear systems solver service that utilizes SOAP arrays of floats. To determine the impact of XML encoding and HTTP transport on the performance of a service,

Part of the gSOAP specification of the MagicSquares service is shown in Figure 3. This algorithm is fast and linear in the size of the matrix. Because the algorithm is fast, it enables us to investigate the XML encoding/decoding and communication latencies rather than the compute latency of the service. The service specification includes the definition of XML base64-encoded matrices in gSOAP. The definitions for SOAP-array matrices and binary DIME attachments that were used in the performance tests are similar (not shown).

The impact of HTTP chunking, compression, base64 encoding, and DIME on the performance of the service. HTTP chunking provides a form of streaming IS MEASURED[21]. Chunking can be used to omit the extra pre-serialization phase that gSOAP uses to determine HTTP content length. Also measured the overhead of real-time HTTP compression to transfer SOAP arrays with gSOAP. The results are shown in Figure 2.
Real-time compression appears to be very expensive but also very effective because it reduces the message length to a size that is comparable to the size of the compressed form of the binary data. The gzip compression rate denotes the length of the original message, and \( c \) denotes the length of the compressed message. Compression is useful when the network bandwidth is limited. However, some experimentation with low-bandwidth networks such as a phone line did not yield any performance gains with compressed HTTP transfers because v.90 modems already apply compression.

Figure 3 (a) shows the response times of the Magic-Squares server using chunked SOAP messages with base64 encoding and SOAP/XML with DIME over the LAN. The startup overhead indicates the network latency to establish a socket connection with the service. gSOAP supports HTTP keep-alive, which means that the startup time can be avoided using HTTP keep-alive connections after establishing a connection with a service. In addition, gSOAP disables the Nagle algorithm to reduce the connection startup latency.

Figure 3 (b) shows the response times of a Java implementation of the Magic Squares server using Java RMI over the LAN (JDK 1.3 with JIT). The startup overhead is the time it takes for the Java RMI lookup operation to establish a connection to the server. The SOAP/XML service response is almost twice as fast with HTTP chunking and DIME compared to the Java RMI implementation.

V. DISCUSSION

This section briefly summarizes the results and discusses these findings in the context of scientific computing with Web and Grid services.

File Transfers File transfer protocols such as FTP and GridFTP are commonly used to exchange large static data sets. However, SOAP/XML with streaming DIME is more flexible than these file transfer protocols. First of all, the SOAP/XML request and response messages can be utilized to carry meta-data such as application-specific information items. Secondly, multiple files can be streamed at once. Thirdly, streaming DIME supports the exchange of dynamic data. In the last, the industry standard WS routing and WS security protocols are orthogonal to SOAP/XML/DIME which means that file transfers based on Web Services can be routed and protected.

Data Encodings SOAP and WSDL are protocols that exploit the richness of the XML schema specification standard for defining XML content. The XML schema extension and restriction features introduce an object oriented approach to data modeling with XML. It enables the definition of derived data types to optimize the exchange of application-specific data formats such as numerical arrays. DIME attachments provide the most efficient means to exchange raw binary data. While binary data transfers are often best utilized for platform-independent streaming media types such as images and sound, binary numerical data exchange is platform dependent due to the differences between big- and little-endian architectures and internal floating point representations. The results presented in this paper demonstrated that the base64 encoding is almost as efficient as DIME transfers of data sets. Base64 has the advantage that individual data items (ints and floats) can be encoded within the SOAP/XML message contents such as SOAP arrays, for example.

HTTP Keep-Alive A client can request a persistent connection with a service using HTTP keepalive. However, the service may drop the connection after a while thereby forcing the client to reconnect. The startup time to establish a connection between a gSOAP client and service is relatively low and less than 0.5 ms (550MHz P3, Linux RH, and 100BaseT LAN)[21]. HTTP keep-alive is enabled in gSOAP with the command \texttt{soap.set_mode(soap, SOAP\_IO\_KEEPALIVE)}.

HTTP Chunking HTTP chunked transfers can be utilized either by a client or a service or both. HTTP chunking is a simple form of streaming, because the chunked SOAP/XML messages are not buffered to determine the HTTP content length header. In fact, in contrast to other SOAP implementations, gSOAP does not buffer a message to determine the HTTP content length header, but uses a two-phase serialization scheme consisting of a message length counting phase followed by the transmission of the message. This scheme is implemented by serializing the message twice. HTTP chunking limits the message transmission to only one phase, which is faster. In addition, gSOAP implements a form of latency hiding with chunked transfers, where communication is overlapped with the serialization computations. HTTP chunking is enabled in gSOAP with the command soap.set_mode.

HTTP Compression should be used carefully. The overhead of the Zlib compression algorithm outweighs the benefit of the message size reduction to improve bandwidth utilization, for example. Most modems and some network routers already compress TCP/IP packets on the fly, which eliminates the need to compress SOAP/XML messages. However, compression is still useful to store data in compressed XML using gSOAP’s data serializers. HTTP gzip compression is enabled in
VI. CONCLUSION

We have shown that it is impossible to reliably provide atomic, consistent data when there are partitions in the network. It is feasible, however, to achieve any two of the three properties: consistency, availability, and partition tolerance. This paper discussed the use of SOAP/XML Web and Grid Services for scientific computing. SOAP and WSDL are based on extensible schemas which allow the selection of alternative XML layouts for application data depending on interoperability and performance goals. Performance testing with HTTP chunking, compression, binary DIME attachments, and XML representations such as SOAP Arrays and base64 shows that significant performance gains with SOAP/XML Web Services are possible, surpassing the performance of Java RMI, for example. Formalizing this idea and studying algorithms for achieving it is an interesting subject for future theoretical research.

REFERENCES

[21] Pushing the SOAP Envelope With Web Services for Scientific Computing by Robert A. van Engelen, School of Computational Science and Information Technology, Florida State University, Tallahassee.