Efficient Processing of XML Documents

Girish Tere
Department of Computer Science,
Shivaji University,
Kolhapur 416 004, India

Bharat Jadhav
Department of Electronics and Computer Science,
Y.C. Institute of Science,
Satara – 415 001, India

ABSTRACT

XML is now worldwide standards for data definition. The standard of data transfer and exchange between organizations through Web services has become exceedingly popular, especially in electronic commerce. Web services are somewhat loosely defined, but may be characterized in general as using existing web technologies and standards to build the distributed computing environments. Data transferred through web services is in the form of XML. XML is universal language for information exchange and has been used by many organizations for developing enterprise applications. With the widespread adoption of SOAP and Web services, XML-based processing, and parsing of XML documents in particular, is becoming a performance-critical aspect of business computing. There are connections between Formal languages, Automata theory and XML. Finite State Machines (FSM) provide a powerful way to describe dynamic behavior of systems and components. XML has many important features, including platform and language independence, flexibility, expressiveness, and extensibility. To improve web service performance, we have parsed XML documents using Deterministic Finite Automata (DFA). DFA is constructed to efficiently parse XML documents containing SOAP messages by encoding the XML parser's states as a DFA. In this paper we discuss our simple application and performance results we obtained.

General Terms

Distributed Computing, Internet Programming

Keywords

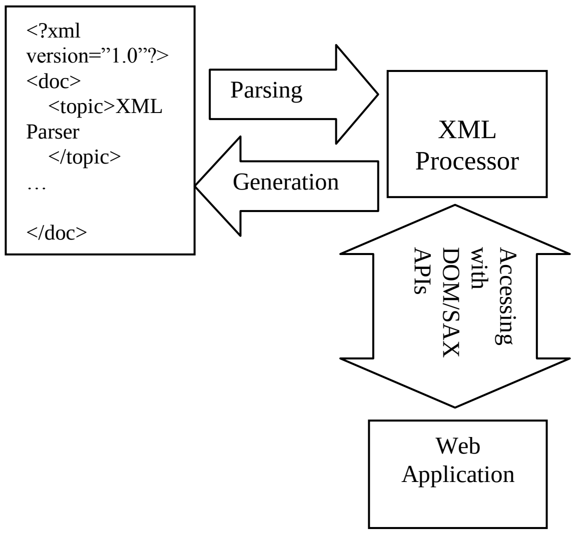
Deterministic Finite Automata, SOAP, WSDL, XML documents.

1. INTRODUCTION

Use of Web services is mandatory in developing enterprise application. Producer of Web services need to publish Web service using WSDL and consumer of Web service need to search the needed Web service and use the Web service as per the contents of WSDL. WSDL documents are basically a contract for using Web service. WSDL plays an important role in the overall Web services architecture since it describes the complete contract for application. Web services communicate to each other as well as with client using WSDL contract defined in XML. This communication is done by exchanging SOAP messages. SOAP messages are basically XML documents. SOAP is an XML-based messaging protocol. It defines a set of rules for structuring messages that can be used for simple one-way messaging but is particularly useful for performing RPC-style (Remote Procedure Call) request-response dialogues. It is not tied to any particular transport protocol though HTTP is popular. SOAP is an important building block for developing distributed applications that exploit functionality published as services over an intranet or the internet. Many studies [7],[8],[9],[11],[15] have shown that the use of XML can lower performance. XML primarily uses UTF-8 as the representation format for data. Sending commonly used data structures via standard implementations of SOAP incurs severe performance overheads, making it difficult for applications to use Web services. Due to the widespread adoption of standards in Web services, it is critically important to investigate the impact on performance for the kinds of XML used in web applications. The flexibility and loose coupling of XML-based standards allows senders and receivers of XML data to independently deploy selected optimizations, according to the communication patterns and data structures in use [6],[13]. SOAP has plus points like transparency, expressiveness, platform and language independence, extensibility and robustness. SOAP is a popular choice as the common underlying protocol for interoperability between Web services. These features facilitate the use of SOAP in diverse applications with widely varying characteristics and requirements. Clients and Web service endpoints can also add optimizations in their implementations. The convergence of Web services standards has made SOAP important, requiring the evaluation of SOAP for data types and communication patterns used by grid applications. It is thus important to have a test framework to determine if a particular SOAP toolkit can meet the performance requirements of an application, or if some other communication protocol should be employed. Finite State Machines (FSM) provide a powerful way to describe dynamic behavior of systems and components. XML has many important features, including platform and language independence, flexibility, expressiveness, and extensibility [18],[19],[21]. Thus, the combination of these characteristics with the interoperability trait of Web services is an attractive way to design distributed applications. For processing of XML documents, we have used DFA based approach, as DFAs are executed faster on any computer [10].

2. GENERATING DFA BASED PARSER

Parsing is the process of reading a document and dissecting it into its elements and attributes, which can then be analyzed. In XML, parsing is done by an XML processor, the most fundamental building block of a Web application [18], [23]. All modern browsers have a built-in XML parser. An XML parser converts an XML document into an XML DOM object - which can then be manipulated with a JavaScript. As the document is parsed, the data in the document becomes available to the application using the parser. This process is shown in Fig. 1. The XML processor parses and generates an XML document. The application uses an API to access objects that represent part of the XML document.



```
<?xml
version="1.0"?>
<doc>
    <topic>XML
Parser
    </topic>
…

</doc>
```

Fig. 1.. XML Parsing Process

We developed WSDL Processor [2],[11], which accepts WSDL or Schema and generates source codes for the implementation of a high-performance Web service as shown in Fig. 2. WSDL is an XML-based language for describing Web services and how to access them. A WSDL document is just a simple XML document. It contains set of definitions to describe a web service.

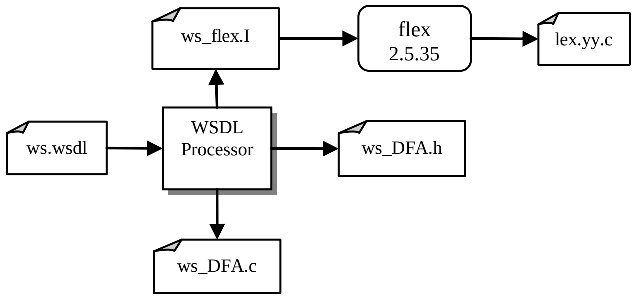


Fig. 2. Parser Genrator

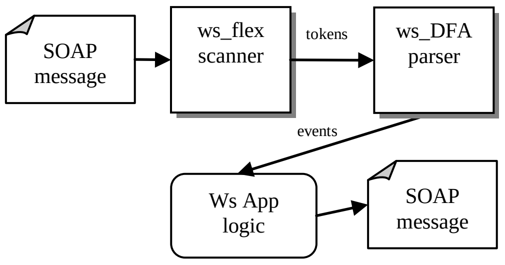


Fig. 3. Processing Web service

Many parsers are built using lexical analysis generators that tokenize input into logical chunks, called as tokens [1],[3]. These tokens act as input for the parser. These generators generally take as input a collection of regular expressions matching each class of token and actions that are executed when those tokens are matched. This is explained in Fig. 3. Some of valid tokens are elements, attributes and data. flex is a commonly used scanner generator. flex transforms the collection of regular expressions into a Non-Deterministic Finite Automaton (NFA) which recognizes the union of the patterns[12]. This NFA is converted into a DFA and then reduced. The DFA is written out as a table of states of two dimensions, DFA state and character input. The input is scanned by reading each character in input order, looking up the current state and current input in the DFA table to find the next state. Any required action is executed when the state changes.

3. APPLICATIONS

Our example describes a parser for a `repeatMessage` service in detail. Fig. 4 shows the schema/DTD of the repeatMessage SOAP request message. The `repeatMessage` message element contains a child element `message` of type XSD string.

```
<schema
targetNamespace="urn:repeatMessage"
xmlns:xsd="http://www.w3.org/2001/XMLS
chema"
xmlns="http://www.w3.org/2001/XMLSchem
a">
<element name="repeatMessage">
<customType>
   <sequence>
<element name="message"
type="xsd:string"/>
<any namespace="##any" lowerlimit="0"
upperlimit="100"/>
   </sequomence>
</customType>
</element>
</schema>
```

Fig. 4. The repeatMessage Message Schema

The wsdlProcessor tool generates the Flex description shown in Fig. 5. For this example, the PUSH and POP operations are simply keeping track of the node nesting level in the document. The level indicator is used for controlling the DFA transitions. The source code of the DFA generated by wsdlProcessor is shown in Fig. 6. The `yylex` function returns the next token from the Flex scanner shown in Fig. 5.

```
%{
#include "repeatMessageDFA.h"
#define PUSH level++;
#define POP level--;
%}
blank [ \t\v\n\f\r]*
name [^>/:= \t\v\n\f\r]+
qual {name}:|""
open <{qual}
close [^>]*>
data [^<]*
```

```
%%
{blank}               // ignore white space
{open}"?"{close}      // ignore declaration
{open}"!"{close}      // ignore comment
{open}"/"{close}      POP
{open}"Header"{close} PUSH return HEADER;
{open}"Body"{close} PUSH return BODY;
{open}"repeatMessage"{close}    PUSH    return
ELEMENT_repeatMessage;
{open}"message"{close}      PUSH      return
ELEMENT_message;
{open}{name}"/"{close}
{open}{name}{close}         PUSH
{data}                      return DATA;
<<EOF>>                     return EOF;
%%
```

Fig. 5. Flex Specification for repeatMessage

Fig. 6 shows the source code generated by WSDL Processor [2], [10],[14]. The `event("repeatMessage/message", yytext)` call returns an XPath expression and string data to the server application.

```
int repeatMessageDFA()
{
   int token, state = 0;
   while ((token = yylex()) != EOF)
   {
         switch (state) {
            case 0: if (token == BODY && level == 2)
                          state = 1;
                    break;
            case 1: if (token == ELEMENT_repeatMessage &&
                   level == 3)
                          state = 2;
                    break;
            case 2: if (token == ELEMENT_message && level
                   == 4)
                          state = 3;
                    break;
            case 3: if (token != DATA || level != 4)
                          return error("Invalid input
                            value");
                    event("repeatMessage/message",
                            yytext);
                    state = 4;
                    break;
            case 4: if (token == EOF && level == 0)
                          return ACCEPT;
                    return error("Invalid message");
          }
   }
   return error("End of file");
}
```

Fig. 6. DFA for repeatMessage

4. PERFORMANCE ANALYSIS

Experiment were carried on Dell Inspiron Laptop with Intel Core 2 Duo, 4 GB RAM. We measured Web services performance on the same machine, i.e. client and server was installed on the same machine. The performance of the repeatMessage application is compared to the performance of parser built with gSOAP 2.5.1 and the performance of eXPAT parser [11],[26]. This paper shows the performance of the same repeatMessage application with gSOAP 2.5.1 parser and the performance of Xerces2 XML parser [20]. The performance of the Xerces2 parser is considered to be good. Xerces2 Java is a library for parsing, validating and manipulating XML documents. The gSOAP toolkit is an efficient implementation of Web services standards in C and C++ [16], [17],[26]. We changed the message string size n from 256, 512 to 1024 and repeated the experiment. Performance comparison of these parsers with varied n is shown in Fig. 7, 8 and 9. Performance analysis of the result obtained show that the performance of DFA-based parser is better than other two parses considered, viz. gSOAP and Xerces2. We have measured the response time using Apache Jmeter software.

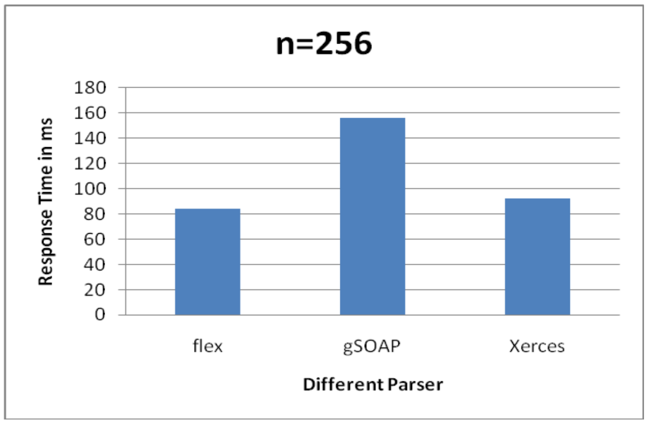


Fig. 7. Performance of parsing repeatMessage application with n=256

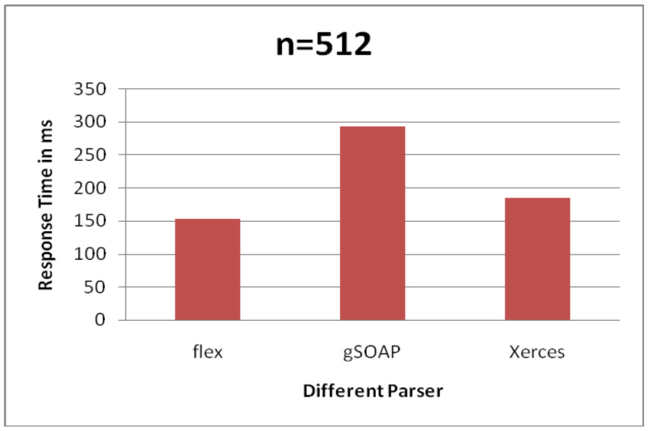


Fig. 8. Performance of parsing repeatMessage application with n=512

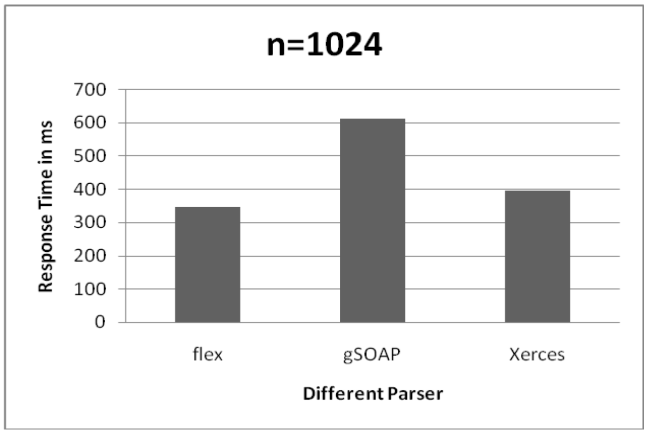


Fig. 9. Performance of parsing repeatMessage application with n=1024

5. CONCLUSIONS

Many researchers tried to improve XML parsing. Web services interact with client by exchanging SOAP messages and can be used as per WSDL. Both SOAP and WSDL are basically XML document. Therefore to improve Web services performance, we tried to improve XML parsing work. XML parser has some finite states. Therefore, we used DFA to effectively reduce computational overheads for XML parsing of SOAP/XML messages. The DFA is generated by a code generator that takes a WSDL as input and generates codes for an optimized schema-specific SOAP/XML message parser. We observed that the performance of the DFA-based parser built with a scanner produced with Flex is better than the performance of parser built with gSOAP 2.5.1 and Xerces2 parser. However, we have not considered processing of XML namespaces. For developing this parser, we considered only ordered XML documents.

6. REFERENCES

[1] A. Aho, R. Sethi, and J. Ullman, “Compilers: Principles, Techniques and Tools”, Addison-Wesley, 2nd ed, 2006.

[2] Aaron Skonnard, Understanding WSDL, Microsoft Corporation, 2010, http://msdn.microsoft.com/en-us/library/ms996486(printer).aspx, Accessed on 30th Aug 2010

[3] Abu-Ghazaleh N., Govindraju M., Lewis M. J., Optimizing performance of web services with chunk-overlaying and pipelined-send. Proceedings of the International Conference on Internet Computing (ICIC), June 2003, 482–485.

[4] Apache Software Foundation, Xerces2 Java Parser, http://xml.apache.org/xerces2-j, Accessed on 23 July 2010

[5] A. Slominski, XML Pull Parser version 2.1.8.,

[6] C. Chan, P. Felber, M. Garofalakis and R. Rastogi, “Efficient Filtering of XML Documents with XPath Expressions”, In Proceedings of the International Conference on Data Engineering, 2002.

[7] C. Kohlhof and R. Steele, “Evaluating SOAP for high performance business applications: Real-time trading systems”, In proceedings of the 2003 International WWW Conference, Budapest, Hungary.

[8] D. Davis and M. Parashar, “Latency performance of SOAP implementations”, In proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid, 2002.

[9] Danny Chen, Raymond K. Wong, “Optimizing The Lazy DFA Approach for XML Stream Processing”, The Fifteenth Australasian Database, Conference (ADC2004), Dunedin, New Zealand, Vol. 27, 2004

[10] F. Neven, “Automata theory for XML researchers”, SIGMOD Record, 31(3), 2002

[11] Girish Tere, Bharat Jadhav, Improving Performance of XML Web Services, ICTSM 2011, CCIS 145, pp. 91–98, 2011, Springer-Verlag Berlin Heidelberg 2011

[12] K. Chiu, W. Lu, “Compiler-based approach to schema-specific XML parsing”, First International Workshop on High Performance XML Processing, New York, USA, May 17–22, 2004, ACM Press, 2004.

[13] K. Chiu, M. Govindaraju, and R. Bramley, “Investigating the limits of SOAP performance for scientific Computing”, In proceedings of the 11th IEEE International Symposium on High-Performance Distributed Computing, 2002.

[14] Lei Li, Chunlei Niu, Ningjiang Chen, Jun Wei, "High Performance Web Services Based on Service-Specific SOAP Processor", IEEE International Conference on Web Services (ICWS'06), 2006, pp 603-610

[15] M. Murata, D. Lee, and M. Mani, “Taxonomy of XML schema languages using formal language theory”, In Extreme Markup Languages, 2001.

[16] R. van Engelen, “Pushing the SOAP envelope with Web services for scientific computing”, In proceedings of the International Conference on Web Services (ICWS), pages 346–352, Las Vegas, 2003.

[17] R. van Engelen and K. Gallivan, “The gSOAP toolkit for web services and peer-to-peer computing networks”, In 2nd IEEE International Symposium on Cluster Computing and the Grid, 2002.

[18] R. van Engelen, G. Gupta, and S. Pant, “Developing web services for C and C++”, IEEE Internet Computing, March 2003, pp 53-61

[19] T. Green, G. Miklau, M. Onizuka, D. Suciu, “Processing XML Streams with Deterministic Automata”, 9th International Conference on Database Theory, Siena, Italy, 8-10 January 2003.

[20] The Apache Xerces Project, http://xerces.apache.org/, Accessed on 17th March 2011

[21] Welf M. Löwe, M. L. Noga and T. S. Gaul, ‘‘Foundations of Fast Communication via XML”, Annals of Software Engineering 13, Nos. 1–4, 357–359 (June 2002).

[22] Wei Zhang, Robert A. van Engelen, “An Adaptive XML Parser for Developing High-Performance Web Services”, Fourth IEEE International Conference on eScience, 2008, pp 672-679

[23] Wei Zhang van Engelen, R.A., “High-Performance XML Parsing and Validation with Permutation Phrase Grammar Parsers”, ICWS '08. IEEE International Conference on Web Services, 2008, Beijing, pp 286 – 294

[24] Wim Martens, Joachim Niehren, On the minimization of XML Schemas and tree automata for unranked trees, Journal of Computer and System Sciences, Vol 73 Issue 4, June 2007

[25] XMLTK, The XML toolkit, http://www.cs.washington.edu/homes/suciu/XMLTK/, University of Washington, 2002, Accessed on 25 Aug 2010

[26] Wichaiwong, T. Jaruskulchai, C., A Simple Approach to Optimize Web Services' Performance, Third International Conference on Next Generation Web Services Practices, NWeSP, Seoul 2007, pp 43-48