Web Service Performance using SOAP Compression

G M Tere  
Department of Computer Science,  
Shivaji University, Kolhapur,  
Maharashtra - 416004, India  
girish.tere@gmail.com

R. R. Mudholkar  
Department of Electronics,  
Shivaji University,  
Kolhapur,  
Maharashtra – 416004, India  
rrm_eln@unishivaji.ac.in

B T Jadhav  
Y.C. Institute of Science,  
Satara,  
Maharashtra - 415001, India  
bjt21875@gmail.com

Abstract—Recent distributed computing applications are developed using different platforms working on different Operating Systems and these applications are running on heterogeneous hardware. SOAP protocol is mainly used for communicating among heterogeneous processes developed in platforms like .Net, Java, Android and so on. SOAP messages are basically XML documents and thus because of use of XML messages for interprocess communication there is a chance that performance of distributed computing applications will suffer. Client’s request and server’s response is sent in SOAP format and the size of SOAP message may be large if it is carrying information from databases. To transport such a large SOAP message more time will be required resulting in increase in response time and decrease in throughput. One solution is to use run time compression technique to compress SOAP message. Our experiments show that response time is reduced by 56% while using compression technique.

Keywords—Compression, performance, SOAP, SOAP Extension, Web service

I. INTRODUCTION

On more complex XML Web services, the SOAP response could be a very large dataset. For instance, when it is serialized to XML, a dataset that contains all columns of the table "orders" from Northwind database is about 454 KB. We developed an application that retrieves this dataset by invoking an XML Web service; the SOAP response would contain all of that data. Web services describe a standardized way of integrating web based applications using the XML, SOAP, WSDL and UDDI techniques. They are used for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other’s IT systems behind the firewall. Web service is suitable for distributed system because it use SOAP protocol. SOAP a simple and extensible for exchange of message, is the most widely used communication protocol in the web service. Beside of advantage, web services have basic problem and that is low performance [7]. Some of approach introduced to improve performance. In this paper, attempt is made to improved performance of web services using runtime compression technique [8].

II. SOAP EXTENSIONS

SOAP Extensions allow us to plug into the SOAP architecture of the .NET Framework in order to perform some pre or post-processing on SOAP messages. When a SOAP message is received by the Web Service's HTTP handler [1], the SOAP message is de-serialized into objects (for example, SoapHeader for header information and SoapMessage for the body), and eventually passed into some Web Method. After the Web Method has finished, the result as well as any SoapHeader, SoapMessage, or SoapException objects are serialized into a SOAP message [5], [6] and sent back up the chain and to the client. SOAP extensions are a Microsoft ASP.NET Web method interception mechanism that can be used to manipulate SOAP requests/responses before they are sent on the network. Developers can write code that executes before and after the serialization and deserialization of messages. SOAP extensions provide a low-level API for implementing serialization and deserialization of a SOAP message.

As SOAP message is basically XML document, it contain lot of redundant data. Therefore we selected compression technique to compress the SOAP response. Using SOAP extensions, we can reduce the size of SOAP messages traveling on the network when a consumer invokes a method from an XML Web service. In most cases SOAP requests are much smaller in size than SOAP responses (for example, a large dataset returned by server), therefore we have compressed only SOAP responses. We used mechanism called object serialization where an object is represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object. The reverse method is called as deserialization. As shown in Fig. 1 on server side and AfterSerialize stage the SOAP response is compressed (shown as ‘1’ in Fig. 1) and travels through network as a compressed SOAP message and on client side and BeforeDeserialize stage the compressed SOAP message is decompressed (shown as ‘2’ in Fig. 1). After uncompressing SOAP message deserialization process is performed.

SOAP extensions must be running on both client and server. If the size of SOAP message is large, the sending of data will be difficult because of limited bandwidth. In presented approach before sending SOAP message in media, message should be compressed in order to decrease transfer time. An important point in compressing of SOAP message is that a whole of SOAP message is not compressed but only the body of SOAP message is compressed, because the header of SOAP message is required for correct function.
III. EXPERIMENT SETUP

- Server side: DELL Inspron N5110 Intel Core i5, 4GB RAM and Windows 7 OS
- Consumer side: Dell Inspiron N51545 Intel Core duo with 4 GB RAM and Windows XP.
- Database used: Microsoft SQL Express Server 2005
- IDE used for application development: Visual Studio 8
- Both laptops were connected using Wi-Fi router.

To compress Web service’s SOAP responses, following steps are used:

- Compress the SOAP response message after serialization on the server.
- Decompress the SOAP response message before deserialization on the client.

The application uses SOAP extension. We developed XML Web service that returns a large dataset created from Orders table of Northwind database. Structure of Orders tables is shown in Fig. 2. The developed application using Microsoft Visual Studio 8 is shown in Fig. 3.
Figure 3. Application developed in .NET to demonstrate use of compression technique.

Fig. 4 shows running of Web service ‘Service’, which contains ‘getorders’ method.

Figure 4. Web service ‘Service’
Invoking `getorders` method give output as shown in Fig. 5.

On the consumer side, we developed a Microsoft Windows application that invokes the above XML Web service, retrieves the dataset, and can be presented in proper UI or can be processed further as per requirement.

The consumer invokes two identical XML Web services, only one of which uses SOAP compression. The Timer class below is used to measure invocation time:

```vbnet
Public Class ClsTimer
    Private Declare Function timeGetTime Lib "winmm" () As Long
    Private lngStartTime As Integer
    Private lngTotalTime As Integer
    Private lngCurTime As Integer

    Public ReadOnly Property TotalTime() As String
        Get
            TotalTime = lngTotalTime
        End Get
        End Property

    Public Sub StartTiming()
        lngTotalTime = 0
        lngStartTime = timeGetTime()
    End Sub

    Public Sub StopTiming()
        lngCurTime = timeGetTime()
        lngTotalTime = (lngCurTime - lngStartTime)
    End Sub
End Class
```

On the server side, the SOAP response from the server is compressed to reduce its size. Following steps are completed for compressing SOAP message at runtime.

1. Using Microsoft Visual Studio .NET, we created a new Microsoft Visual Basic .NET class library project which is extended from “SoapExtension” class.

2. We added the `ServerSoapExtension.dll` assembly as reference and declare the SOAP extension on `web.config` of the XML Web service:

```xml
<?xml version="1.0" encoding="utf-8" ?>
```
We used a temporary folder ("c:\temp"), to capture as text file ("c:\temp\server_soap.txt") the SOAP request and the compressed SOAP response. The content of server_soap.txt are shown in Fig. 6.

![server_soap.txt](image)

Figure 6. Content of text file, server_soap.txt

On the consumer side, the SOAP response from the server is uncompressed to retrieve the original content. To achieve this following steps are completed:

1. Using Visual Studio .NET, we created a new Visual Basic .NET class library project.
2. We added the ClientSoapExtension.dll assembly as reference and declare the SOAP extension on the XML Web service reference of our application:

   ```xml
   <soapExtensionTypes>
     <add type="ServerSoapExtension.myextension, ServerSoapExtension" priority="1" group="0"/>
   </soapExtensionTypes>
   ```

IV. RESULT ANALYSIS

On the consumer side a Windows application invokes an XML Web service. The XML Web service returns a dataset and can be shown on the consumer side. Response time required for sending normal SOAP response message and compressed SOAP message is shown in Fig. 7.

![Form1](image)

Figure 7. Application which invokes the same XML Web Service with and without using SOAP Compression.
As shown in Fig. 7, we need 56% less time in retrieving the dataset, containing information of 800 orders, on the consumer side with very little impact on the CPU load. SOAP Compression can significantly increase the XML Web Services' performance when consumers and servers exchange large data [2]. There are a lot of solutions for improving performance on the web. SOAP Extensions is the way to improve XML Web Services' performance by compressing the exchanged data with small impact on CPU load [5]. We repeated the experiment by varying number of records in an orders table. Results obtained are shown in Table 1 and graphically in Fig. 8. From the experiments performed we conclude that technique of compressing SOAP response is useful when large data is need to be sent. For data less than 10 KB, data should not be compressed. For compressing SOAP we have used SharpZip library [9], which is open source software.

<table>
<thead>
<tr>
<th>Rows in Orders</th>
<th>Time required to send normal SOAP (response in msec)</th>
<th>Time required to send compressed SOAP (response in msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>362</td>
<td>198</td>
</tr>
<tr>
<td>400</td>
<td>396</td>
<td>204</td>
</tr>
<tr>
<td>500</td>
<td>428</td>
<td>215</td>
</tr>
<tr>
<td>600</td>
<td>468</td>
<td>222</td>
</tr>
<tr>
<td>700</td>
<td>517</td>
<td>234</td>
</tr>
<tr>
<td>800</td>
<td>578</td>
<td>250</td>
</tr>
</tbody>
</table>

V. CONCLUSIONS

As SOAP message contain large data, more time is required to transport SOAP message from server to client. This time to transport SOAP message can be reduced if we compress SOAP response message after serialization on the server. SharpZip library is used for compression. We developed .Net application which calls the Web service developed in .Net. If SOAP message is small than 2KB, then compression is not desirable. For larger SOAP size, time required to compress SOAP and uncompress SOAP is negligible as the time required to transport SOAP message. Our experiment show that we have saved nearly 58% of time using SOAP compression at run time.

ACKNOWLEDGMENT

We thank teachers of Department of Computer Science, Shivaji University, Kolhapur for motivating us for this research work. We wish to thank Principal of Thakur College of Science and Commerce, Mumbai and Principal, Y.C. Institute of Science, Satara, for providing resources required to perform experiments and to complete this paper.

REFERENCES


Figure 8. Response time required for sending different data with and without compression