IOSR Journal of Computer Science (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727
PP 12-16
www.iosrjournals.org

Improving Performance of RESTful Web Services

G M Tere[1], R R Mudholkar[2], B T Jadhav[3]
[1](*Department of Computer Science, Shivaji University, Kolhapur, India)*
[2](*Department of Electronics,Shivaji University, Kolhapur, India)*
[3](*Department of Computer Science,Y C College of Science, Satara, India)*

ABSTRACT: *RESTful Web services play important role in distributed web applications. Performance of Web services affects overall performance of applications. Using features of REST, HTTP, and the REST frameworks like Jersey, Restlet, RESTEasy the latency and system resource consumption of application is improved. Performance of RESTful web services is improved using techniques such as fast manipulation of strings, streaming large representations, compressing SOAP response, partial representation, using Caching techniques and using conditional methods. It is observed that performances of RESTful Web services increases by approximately 40% if above techniques are used. This paper describes those techniques for improving performance of RESTful Web services.*

Keywords *-Caching, compression, conditional methods, REST, Streaming*

INTRODUCTION

Using features of REST, HTTP, and the REST frameworks like Jersey, Restlet, RESTEasy [4,5] the latency and system resource consumption of application can be improved. Performance of RESTful web services can be improved using following techniques:

1. Using StringBuilder for handling large number of strings
2. Streaming large representations
3. Compressing SOAP response
4. Partial representation
5. Using Caching techniques
6. Use of Conditional methods

RESEARCH METHODOLOGY

The objective of research work is to improve performance of RESTful Web Services. To verify above mentioned six techniques various experiments are carried out in .NET and Java. Programs are developed and tested with following experiment setup:

Experiment Setup

- Server side: DELL Inspron N5110 Intel Core i5, 4GB RAM and Windows 7 OS
- Consumer side: Dell Inspiron N1545 Intel Core duo with 4 GB RAM and Windows XP.
- IDE used for application development: Visual Studio 10 and Eclipse
- Database used: MySQL
- Both laptops were connected using Wi-Fi router.

Techniques for Improving Performance of RESTful Web Services

A.Using StringBuilder for handling large number of strings

Strings are immutable in the .NET framework as well as in Java. Therefore methods and operations that appear to change the string are actually returning a modified copy of the string. String operations such as append would be more efficient if performed using StringBuilder objects than

String objects. This approach has improved performance lot. If many string manipulations are to be done, one has to use StringBuilder class as it is faster. The time to create a string from 10000 substrings is measured in two different ways. The first time a simple string concatenation is used and the second time the StringBuilder class is used. Output of developed comparison application is shown in Figure 1.

From Fig. 1 it is clear that lot of time can be saved when StringBuilder class is used. In this application strings are handled using StringBuilder class and the time required in string building is reduced by 99% as that of string developed using concatenation.

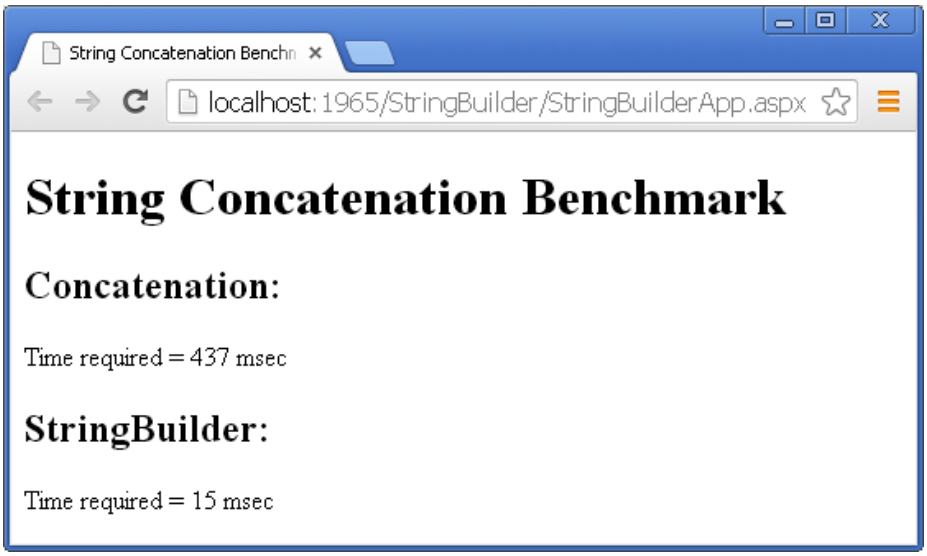

Figure 1: String manipulation using Concatenation and StringBuilder

B. **Streaming representations**

In most use cases, HTTP requests and responses must contain a header that specifies the length in bytes of the message body. For instance, a server that returns a request with the ASCII character string "Good morning!" will include the following header in its response, as the string is composed of 13 characters:

Content-Length: 13

This approach requires knowing the length of the data before sending it. But that's not always the case; for example, the data can be generated dynamically and its length is not known. If the data is large or takes a long time to generate, one might want to start sending it before it is entirely generated. Streaming data like that is made possible by TCP, which is usually the transport protocol use for sending HTTP requests and responses over the network. When a TCP connection is established between the client and the server, data can be sent over that connection in pieces.

In such situations, to solve the unknown Content-Length problem, a technique called content streaming is used. Instead of indicating the entire content length, the entity will contain a series of chunks, using a Transfer-Encoding: chunked header. Each chunk specifies its own size, and the end of the series is marked with a zero-sized chunk.

When Content-Length is not known, a technique called content streaming is used. Instead of indicating the entire content length, the entity will contain a series of chunks, using a Transfer-Encoding: chunked header. Each chunk specifies its own size, and the end of the series is marked with

a zero-sized chunk. Using this technique client receives quick response from server and start processing the data received instead of waiting for full data to be available.

C.Compressing SOAP response

One of the important features of HTTP [3] is its ability to compress the content of entities exchanged between components. This feature relies on the Accept-Encoding and Content-Encoding HTTP headers. REST frameworks like Jersey, Restlet, RESTEasy are capable of automatically compressing and uncompressing entities exchanged with other remote components. For better performance encoding must be enabled.

A major drawback of using SOAP for [10] application integration is its enormous demand for network bandwidth. Compared to classical approaches, like Java-RMI and CORBA, SOAP messages typically cause more than three times the network traffic. In this section compression strategies are explored. Work explains how to use the available network bandwidth effectively. Developed Compress Web service [12] returns a response containing a data from MySQL student table. The response of web services is compressed at runtime using java.util.zip.ZipOutputStream. Output of application is shown in Figure 2.

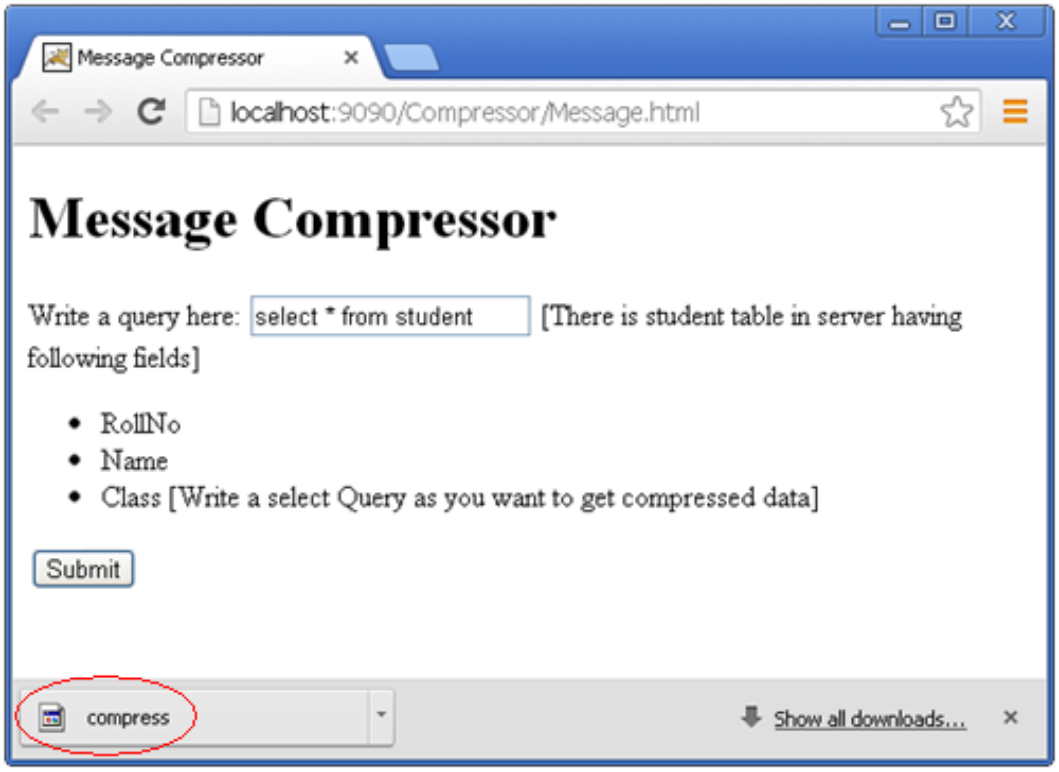

Figure 2: Output of 'Compressor' application, downloading compress.zip file

'compress.zip' file contains temp.log text file. Compress Web service creates a zip file which can sent to client. Thus the response time of a Web service is reduced. The response time of a Web service is measured returning a normal data and Zip data [9]. Results are shown graphically in Figure 3.

Thus available network can be used effectively if large SOAP response is compressed at runtime.

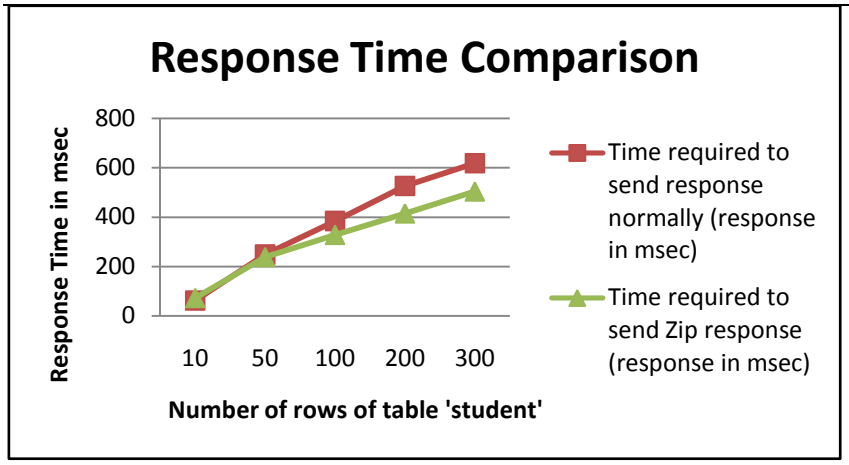

Figure 3: Response time comparison

D. Partial representations

Suppose that, while receiving the representation of some resource in response to a GET request, a client application suffers a temporary network problem that drops the connection to the server. When the network is back and if the representation is large, a server must be able to request only the missing part of the representation instead of restarting from the beginning. A client application may needs a part of a particular representation. In this situation a server must be able to send only the needed resource. For large data sets, this can reduce network traffic local storage capacity requirements and improve performance.Because the server supports the range feature, a client can obtain a specific portion of a resource's representation using the ranges property, which maps to the Range header. Using this technique traffic on network is reduced.

E. Using Caching technique

Caching is one more technique of HTTP performance optimization mechanisms [11]. Caching [1] is a technique in which frequently requested data is stored in memory so that the next time the same information is requested; it can be fetched from memory, rather than asking to application. The purpose of caching is to allow a client to keep previous responses sent by the server and reuse them instead of performing real requests over the network. There are two main benefits of this approach: reduced latency and reduced network traffic. HTTP provides powerful support for caching. If the cached item has expired, either because the underlying data has changed, the time limit has run out, or some dependency has changed, the cache will be invalidated and the next request will retrieve fresh content from the source rather than the cache. This section demonstrates how to add items to the cache. When a client issues a GET request, it can look in its local cache to see if this request has previously been answered by the server and stored in the cache. If this is the case, and if the cached response is fresh enough, the cached response will be used, without the need for network interactions with the server.When implementing a RESTful system [8], caching mechanism is used. Information about the cacheability of responses emitted by server-side application is specified. Caching is a technique in which frequently requested data is stored in memory so that the next time the same information is requested; it can be fetched from memory, rather than asking to application. The purpose of caching is to allow a client to keep previous responses sent by the server and reuse them instead of performing real requests over the network. Using this technique performance of web application is improved.

F. Use of Conditional methods

Conditional methods are used to revalidate cached responses and prevent the lost updateproblem. Sometimes a

client wants to ask the server to process a request only if certain conditions are met. HTTP supports headers that allow conditional method processing [3]:

If-Match, If-Modified-Since, If-None-Match, and If-Unmodified-Since.

A client might want to retrieve the state of a resource (with a GET method), do some processing, and then update the state on the server (with a PUT method) only if the resource state has not changed in the meantime—for example, after concurrent accessing by another client. To achieve this, the client's PUT request should include an If-Match header specifying the entity tag included in the response to the previous GET request.

Conclusion

Web services are used for combining heterogeneous applications. Commercial web applications need to use different heterogeneous applications. Therefore to achieve good performance of these web applications the Web services used in it must perform in efficient manner. Empirical research work conforms that performance of RESTful web services is improved using following techniques:

1. Using StringBuilder for handling large number of strings
2. Streaming large representations
3. Compressing SOAP response
4. Partial representation
5. Using Caching techniques
6. Use of Conditional methods

ACKNOWLEDGEMENTS

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

[1] Caching Guide, Mar 2012
   a. URL: http://httpd.apache.org/docs/2.2/caching.html
[2] ETag, Entity Tag, Jan 2012
   a. URL: http://optimizeasp.net/conditional-get
[3] Gourley D.; Totty B.; Sayer M.; Aggarwal A. and Reddy S. (2002), HTTP: The Definitive Guide, O'Reilly Media
JAX-RS, Java API for RESTful Services, 2011
   a. URL: https://jax-rs-spec.java.net/
[4] Jersey, RESTful Web Services in Java, 2011
   a. URL: https://jersey.java.net/
[5] Mazzetti P.; Nativi S.; Bigagli L. (2008), Integration of REST style and AJAX technologies to build Web applications; an example of framework for Location-Based-Services, Information and Communication Technologies: From Theory to Applications, ICTTA 2008. 3rd International Conference on, vol., no., 1 pp.6, 7-11 April 2008
[6] Meng J.; Mei S.; Zhao Y. (2009), RESTful Web Services: A Solution for Distributed Data Integration, Computational Intelligence and Software Engineering, CiSE 2009. International Conference on, vol., no., pp.1-4, 11-13 Dec. 2009
[7] Richardson L. and Ruby S. (2007), RESTful Web Services, O'Reilly Media
[8] SharpZipLib, Open source compression tool, 2012
   a. URL: http://www.icsharpcode.net/opensource/sharpziplib/
[9] SOAP (2007) Version 1.2. W3C Recommendation (Second Edition), February 2010
   a. URL: http://www.w3.org/TR/soap/
[10] Takase T. and Tatsubori M. (2004), Efficient Web services response caching by selecting optimal data representation. In Proceedings of the 24th International Conference on Distributed Computing Systems, pages 188–197, Toronto, Canada. IEEE Computer Society.
[11] Werner C.; Buschmann C. and Fischer S. (2005), WSDL-Driven SOAP Compression. International Journal of Web Services Research, Vol. 2, Issue 1, pp. 18-35.