ISSN 0975-3303
Mapana J Sci, 9, 2 (2010), 47-52
https://doi.org/10.12725/mjs.13.6

PERFORMANCE ENHANCEMENT OF SOAP VIA MULTI LEVEL CACHING

Samiksha Shukla, D. K. Mishra** and Kapil Tiwari****

ABSTRACT

Due to complex infrastructure of web application response time for different service request by client requires significantly larger time. Simple Object Access Protocol (SOAP) is a recent and emerging technology in the field of web services, which aims at replacing traditional methods of remote communications. Basic aim of designing SOAP was to increase interoperability among broad range of programs and environment, SOAP allows applications from different languages, installed on different platforms to communicate with each other over the network. Web services demand security, high performance and extensibility. SOAP provides various benefits for interoperability but we need to pay price of performance degradation and security for that. This formulates SOAP a poor preference for high performance web services. In this paper we present a new approach by enabling multi-level caching at client side as well as server side. Reference describes implementation based on the Apache Java SOAP client, which gives radically enhanced performance.

Keywords: *SOAP, protocol, XML, caching, (time-to-live) TTL*

* *Corresponding author, Christ University, Bangalore. samiksha.shukla@gmail.com*
** *Acropolis Institute of Technology and Research, Indore*
*** *EMC Software & Services India Pvt. Ltd, Bangalore*

1. Introduction

Soap stands for Simple Object Access Protocol, it is a communication protocol between applications, and SOAP is platform independent as well as language independent. In recent applications communication takes place using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC signifies a compatibility and protection problem; firewalls and proxy servers will normally block this kind of traffic.

An enhanced way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. In order to achieve this SOAP was created. Enabling caching over web gives some benefits like: it decreases network traffic, load on main server and response time.

Simple Object Access Protocol (SOAP) [1] is a recent and emerging technology in the field of web services technology, which aims at replacing traditional methods of remote communications. Basic aim of designing SOAP was to increase interoperability among broad range of programs and environment, SOAP allows applications from different languages and installed on different platforms to communicate with each other over the network. Web services demand security, high performance and extensibility. SOAP provides various benefits for interoperability but we need to pay price of performance degradation and security for that. This formulates SOAP a poor preference for high performance web services.

There has been a remarkable development in the area of web services. SOAP is one such advancement, which was conceived when there was a requirement for a standard. SOAP is based on XML and thus achieves high interoperability when it comes to exchange of information in a distributed computing environment. SOAP, carrying the advantages that increase with XML, has few disadvantages, which restrict its usage. As SOAP requires messages to be in XML format, so processing of these messages takes significant amount of execution time, which is a great overhead in computation of SOAP.

SOAP is proving its bravery in the enterprise applications. Developers use the protocol to combine unrelated enterprise applications, and to develop new distributed and scalable enterprise applications that employ XML for cross-component messaging.

In this paper, we discuss the client and server side processing of a SOAP request and analyses the SOAP request made by the client to the server when it requests a service from it. After using caches how the performance is enhanced. Each of the phases is further examined to find out where the most of the clients time is spent. As SOAP requires messages to be in XML, a typical request from the client involves XML encoding, which is basically serialization and distribution of the payload, before it is sent to the server.

The aim of this research is to give an idea about how SOAP is more efficient to deal with the requirements of a high performance web service, while still satisfying with the SOAP standard. The client side, after analysis of each phase of its implementation, is optimized by using a client-side caching method. It increases the performance enormously by caching the client requests, which are of small size. In some application such as stock watches request message will be identical which result in identical response message for a time interval, in such situation we can do caching at server side by specifying time-to-live (TTL) value, so in cache hit response time improves but in cache miss overhead of cache lookup is found. In section 2 we shall discuss about the related work, section 3 & 4 gives problem definition and proposed architecture description, section 5 gives limitation of proposed architecture and in section 6 we conclude the paper with summary of our work and future enhancement.

Related Work:

In the Java implementation of Apache SOAP 1.2 and the most common model of SOAP that is used in distributed software, the RPC-style, rather than the message-style, which is less popular. This choice is obvious among web developers, as it closely resembles the method-call model.

There have been various studies comparing SOAP with other protocols, mainly binary protocols such as Java RMI and CORBA. All of this research has proven that SOAP, because of its dependency on XML, is inefficient compared to its peers in distributed computing. References [3, 4, 5, 12] explain, from where SOAPs slowness originates and considered various attempts to optimize it.

Reference [6] describes client-side caching strategy for SOAP services using the Business Delegate and Cache Management design patterns.

Problem Definition:

We must consider performance, in order to develop rich Java clients with SOAP-based servers as data providers.

When our client application accesses the same information (such as stock market) frequently from the server, we will promptly recognize the server's performance, and therefore our application's response time. Further, when hundreds of client applications hit the server concurrently, the performance puts down even faster. There we will be need of caching, raising a SOAP call yields costs similar or more, incurred for running an SQL statement in a RDBMS.

The cache could also be placed at client side as well as server side for performance enhancement, and in our implementation we are going to propose an approach where we will use multilevel caching. For every SOAP request message method will be invoked with few parameters which are always needed by the client for invoking web services. The cache then returns with a SOAP reply to the caller, either from the client's cache, by retrieving the message from the Server's cache or by retrieving the message from the server afresh. The Application will remain transparent about the source of the response message. It increases performance, if the cache has a copy of the reply message.

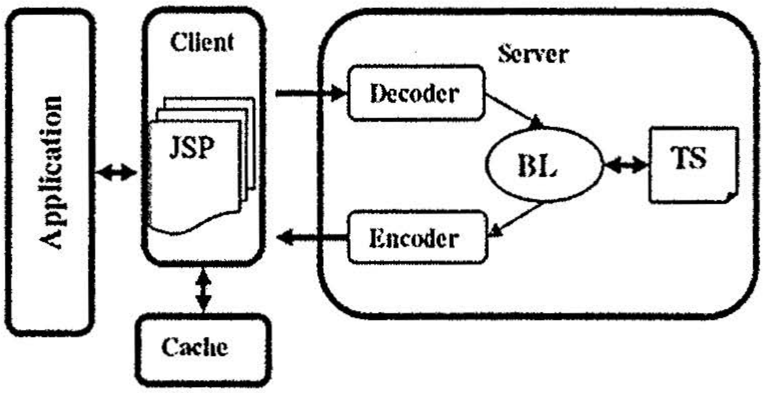

Figure 1: Proposed architecture

In figure1, the proposed architecture we are implementing client side caching as well as server side caching. In this architecture it will store SOAP replies in cache at server side with TTL specification, whenever requirement comes from the application side, web service client creates a SOAP message Dispatch object and calls the invoke method on this object. When cache is supposed to be used then cache invoke method will be called. The cache uses this SOAP Request and the desired web service definition language URI as a key into a table and checks if this entry exists in cache. If it is, then the response can be directly returned by the cache. If not, a cache miss occurs and the actual web service is invoked. When we invoke actual web service request will be received at server side, and server side cache is consulted. If a response message exist and TTL of the service has not passed, then cached message is returned from server. Otherwise the process to service request is executed and the returned message is cached at server side for forthcoming requests from any of the client in the network.

In our architecture, the cache is implemented using Entity Beans and a Frontage (Facade Bean) that brings out the invoke() method. When called, the invoke() method, the SOAPRequest message is canonicalized first and then does an additional function that we call as "compact" which removes superfluous whitespace in the request. We then combine this string message with the web service definition language URI and calculate the Hash key for this combination. The result of the hash function is an input for the Entity Bean's find method. If the key is found, it surely contains the pertinent reply in cache. However, this reply may have become old. In order to assure how old the reply is, we have to compare the current time with the TTL value stored in addition to the SOAPReply. If either the reply was not found in the cache or if the cached entry was old, cache sends a real call to the web service and retrieves the SOAPReply and also examines the WSDL file for the specified operation and checks if the operation enables caching or not. If operation is cacheable then the reply is stored in the cache. In both the cases, the reply is returned to the client. Even if the Web Service is presently disconnected from the client, but if the operation is cached with the cache, it remains available to the client till the cached copy dies.

Limitations:

In order to process in SOAP, message should be in XML, and processing of SOAP message entail significant amount of execution time, which is the utmost overhead in computation of SOAP call. The basic idea behind caching was considered with the belief that SOAP request from the client remains same in most of the instances. The primary requirement is that clients should have fixed set of requests that they can make to the server. Because if it is more, then for each request, the SOAP response will be saved in the cache, increases the size of cache.

References

1. Sun Microsystems. J2ME Web Services Technical White Paper, July 2004. http://java.sun.com/j2me/reference/whitepapers/Web_Svcs_wp072904.pdf.

2. D. Box et al. "Simple Object Access Protocol 1.1", Technical Report, W3C, 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

3. Devaram Kiran, Andresen Daniel, *"SOAP Optimization via Client-side Caching"*, ICWS03, http://people.cis.ksu.edu/~dan/despot/icws03.pdf

4. F. E. Bustamante, G. Eisenhauer, K. Schwan, and P. Widener, "Efficient wire formats for high performance computing", In *Proceedings of Supercomputing 2000*, pages 64–64, 2000.

5. D. Davis and M. Parashar. "Latency performance of SOAP implementations", In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster computing and the Grid*, pages 407–412, 2002.

6. 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 HPDC-11 002 (HPDC'02), page 246. IEEE Computer Society, 2002.

7. O. Azim and A. K. Hamid, "Cache SOAP Services On The Client-Side", http://www.javaworld.com/ javawo rld/jw-03-2002/jw-0308-soap.html?, March,2002.

8. K. Devaram and D. Andresen. "SOAP optimization via parameterized client-side caching". In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2003)*, pages 785–790, Marina Del Rey, CA, Nov. 2003

9. Roy Friedman, "XML Web Service Caching Strategies", MSDN,http://msdn.microsoft.com/ enus/library/ aa480499.aspx Caching web services in mobile ad-hoc networks: opportunities and challenges, ACM Workshop

10. On Principles of Mobile Computing, 2002

11. Kamal Elbashir, "Transparent Caching of Web Services for Mobile Devices ", http://citeseerx.ist.psu.edu/

12. viewdoc/summary?doi=10.1.1.2.1608

13. Samiksha Shukla, Kapil Tiwari "A Research Perspective: SOAP Optimization with Client Side Caching", In *Proceedings* of International Conference on Information Technology and Business Intelligence (ITBI'09).

14. Soaplet, http://www.codehaus.org/ dandiep/soaplet/, June 2004.