High Performance Approach for Server Side SOAP Processing

Lei Li, University of Science & Technology of China, and Chinese Academy of Sciences, China
Chunlei Niu, Chinese Academy of Sciences, China
Ningjiang Chen, Guangxi University, China
Jun Wei, Chinese Academy of Sciences, China
Tao Huang, University of Science & Technology of China, and Chinese Academy of Sciences, China

ABSTRACT

Based on XML, Web services inherit not only the advantages of XML, but its relatively poor performance, which makes it a poor choice for many high-performance applications. In this paper, we propose a new approach to improve Web services performance. Focusing on avoiding traditional XML parsing and Java reflection at runtime, this paper develops a service-specific SOAP Processor to accelerate execution. Moreover, SOAP Processor embeds several cache implementations and uses a novel adaptive cache mechanism, which can choose an optimized cache implementation dynamically in the runtime. Through our experiments in this paper, we observed that our approach can achieve a huge performance gain by incorporating the SOAP Processor into the SOAP engine. [Article copies are available for purchase from InfoSci-on-Demand.com]

Keywords: Caching; Deserializing; High Performance; SOAP Processor; Web Services; XML Parsing

INTRODUCTION

Currently, there has been a tremendous development in the area of Web services and the simple object access protocol (SOAP) is a dominant enabling technology in it. Over the last few years, a lot of works (e.g., Bustamante, 2000; Chiu, Govindaraju, & Bramley, 2002; Elfwing, Paulsson, & Lundberg, 2002; Kohlhoff & Steele, 2003) have been carried out in comparing SOAP with binary protocols, such as Java RMI and CORBA. These researches show that there is a dramatic difference in the amount of encoding necessary for data transmission, when extensible markup language (XML) is compared with the binary encoding style followed in CORBA, and all these researches have proven that SOAP, because of its reliance on
XML, is inefficient compared with its peers in distributed computing.

Although the performance of Web services has been adequate for many important purposes, processing speed unfortunately remains a problem in more performance-sensitive applications, especially when the SOAP engine is built in Java. Existing researches indicate that the performance of those systems built in Java cannot be compared to the system built in C/C++. This is because the C++ compiler translates source code into an executable file, which can be executed by an operation system directly, while Java source code is interpreted into bytecode by Java virtual machine. Eckel (2003) indicates that interpreted Java runs in the range of 20 times slower than C. Hence, the Java-based systems need to improve performance, particularly.

Many performance problems of Web services are inherent in core features of XML: it is text based, flexible in format, and carries redundant information. Let us simply describe the message processing flow in the SOAP engine in general. First, when the SOAP engine receives the message, it uses XML parser to parse the message. Second, it translates the parsed XML elements into Java objects by using deserializing operations. And then, it invokes service business logic with those objects. After that, it gets the results and translates them into a response message by using serializing operations. Finally, the response message will be sent to the user. Davis and Parashar (2002) indicate that in the overall processing flow, the XML parsing and Java reflection at runtime is exactly the performance bottleneck of Web services. Obviously, if we can avoid the time-consuming operation in the runtime, the performance may be greatly improved. There are two intuitive ways to solve this problem, one is to use a cache mechanism to avoid time-consuming operations, and the other is to deal with them before runtime.

Aiming at the problems, this article proposes a novel approach to improve the performance of Web services by optimizing SOAP processing. The SOAP processor presented in the article maintains specific SOAP processors for each Web service. The SOAP processor is generated by using the information from WSDL and service’s classes during system startup or service hot-deployment. In essence, the SOAP processor is a composition of SOAP parser and deserializer. It can identify the SOAP messages related to a specific Web service. Moreover, the SOAP processor embeds several cache implementations and uses an adaptive mechanism to choose the optimized caching strategy dynamically. When receiving a message, the SOAP processor first uses an adaptive caching mechanism to choose the best cache implementation to handle the message. If the cache fails, it uses XML a preparsing framework and predeserializing templates to complete the processing. Using a service-specific SOAP processor we can avoid traditional XML parsing and Java reflection at runtime. Moreover, because the SOAP processor scans a SOAP message only once, the performance is improved remarkably. Our method supports the WSDL document/literal style, which is the direction where the industry is headed; in addition, it is the most consistent model from a Web services architecture perspective (Curbera, Duftler, & Khalaf, 2005). The evaluation presented in this article shows our approach is effective in both speed and memory usage aspects.

The main contributions of this article are as follows: (1) a novel approach to preparse and predeserialize the SOAP message; (2) an adaptive caching mechanism to dynamically choose an optimized cache implementation at runtime; (3) a novel service-specific SOAP processor based on the combination of above two approaches, which can help the SOAP engine to accelerate message processing.

The rest of the article is organized as follows: First, we survey the related works in the following section. We then analyze the performance problem. The forth section describes the preparsing and predeserializing approach and in the fifth section we present the hit ratio-based adaptive caching mechanism and introduce the implementation. The sixth section shows the results of experiments and finally, we conclude...
Related Content

Matchmaking for Business Processes Based on Choreographics
[www.irma-international.org/article/matchmaking-business-processes-based-choreographics/3048/](www.irma-international.org/article/matchmaking-business-processes-based-choreographics/3048/)

Benefits and Challenges for BPM in the Cloud
[www.irma-international.org/chapter/benefits-and-challenges-for-bpm-in-the-cloud/217909/](www.irma-international.org/chapter/benefits-and-challenges-for-bpm-in-the-cloud/217909/)

Bandwidth and Latency Considerations for Efficient SOAP Messaging
[www.irma-international.org/article/bandwidth-latency-considerations-efficient-soap/3074/](www.irma-international.org/article/bandwidth-latency-considerations-efficient-soap/3074/)

Big Data and Innovation in the Delivery of Public Services: The Case of Predictive Policing in Kent

Analysis of Grievances in the Banking Sector through Big Data
[www.irma-international.org/chapter/analysis-of-grievances-in-the-banking-sector-through-big-data/217935/](www.irma-international.org/chapter/analysis-of-grievances-in-the-banking-sector-through-big-data/217935/)