Performance Issues Related to Web Service Usage for Remote Data Access

V.F. Pais, V. Stancalie, F.A. Mihailescu, M.C. Totolici

NationalInstitute for Laser, Plasma and Radiation Physics, Lasers Department
Str. Atomistilor, 409, PO Box MG-36, 077125, Magurele, Romania
Association EURATOM/MEdC

Abstract. Web services are starting to be widely used in applications for remotely accessing data. This is of special interest for research based on small and medium scale fusion devices, since scientists participating remotely to experiments are accessing large amounts of data over the Internet. Recent tests were conducted to see how the new network traffic, generated by the use of web services, can be integrated in the existing infrastructure and what would be the impact over existing applications, especially those used in a remote participation scenario.

Keywords: web services, remote data access, bandwidth consumption.
PACS: 07.05.Bx, 07.05.Rm

INTRODUCTION

An increasing number of applications are starting to use Web Services for data transfer and remote resource access [1]. This is happening because Web Services provide a system-independent format for transferring data, known as Extensible Markup Language (XML) [2]. This is complemented by a Web Services Description Language (WSDL) [3] file, containing the available methods for accessing the service and the detailed definition of its data structures, usually using XSchema [4] format.

This means that each time an application is using a web service several network connections must be established. Firstly, there is the need to transfer the WSDL file and any external data definitions. Then, the application is invoking the necessary service methods in order to get the desired outputs. Finally, the outcome of computations may be stored in a database for future access.

One of the objectives of research based on small and medium scale devices, such as small tokamaks, is the development of techniques for remote access to data. One such technique is provided by web services. Data sources hosted in networks separated geographically require large transfers over the Internet. This new traffic needs to be integrated in the existing network infrastructure, with as little impact as possible over other network services.

Several tests were conducted to find the impact of web service related traffic.

WEB SERVICE TESTS

In order to measure the overhead introduced by web services for data transfer, a test network [Figure 1.] has been constructed. To simulate a real network environment, various traffic generators [5] were used.

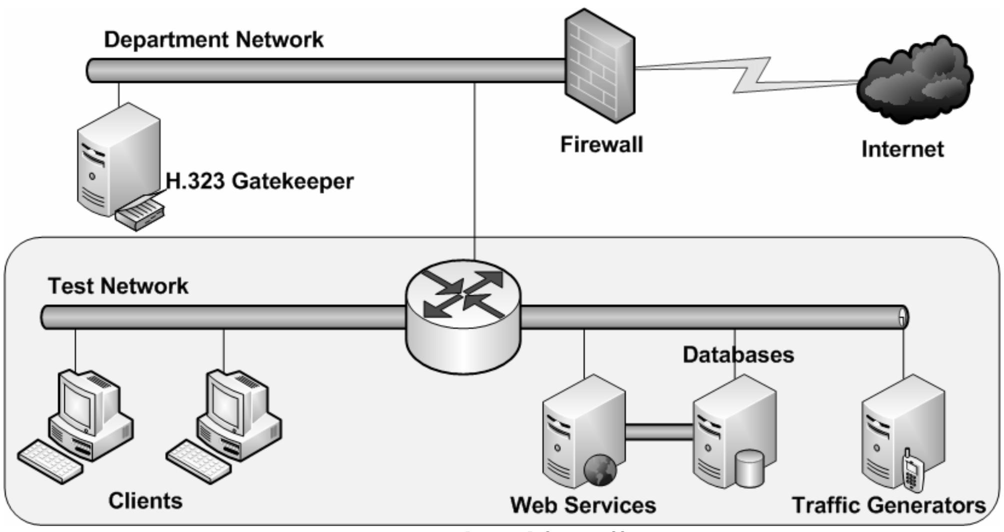

FIGURE 1. Test network used for traffic measurements.

The web services are hosted on a dedicated machine, having a direct link to the database server. The machines are equipped with two Xeon processors, running at 3 GHz, and 4 Gb of memory. The network connectivity is achieved through a copper-based network interface card, with a speed of 1 Gbps.

As a database management system (DBMS) was chosen Postgresql [6]. This choice is justified by the usage of this DBMS in the real network.

Various test scenarios were chosen in order to illustrate the variations in bandwidth usage encountered when data is accessed through web services.

The first series of tests concentrated on the additional information introduced due to the XML encapsulation of data. A generic service was created that would simply expose the underlying data, in existing databases, in order to allow a comparison with direct database requests. Some results are given in [TABLE 1.]. For all these tests, the WSDL associated with the web services had a size of 3691 bytes.

TABLE 1. Size differences between data accessed directly and through web service invocations.

Direct Access (bytes)	**Web Service Access (bytes)**	**Size increase (without WSDL) (%)**	**Size increase (with WSDL) (%)**
6076	22733	274	335
627	1769	182	771
3149	8836	181	298

These results clearly show that XML encapsulation adds a large volume of information to the existing data.

Since web service messages are transferred over Hyper Text Transfer Protocol (HTTP) [7], it is possible to activate compression at HTTP level to reduce the amount of transferred data. However, this is useful only for requests returning a large volume of data. For the tests presented in [TABLE 1.], compression would make a difference only for the first example. The response would be reduced to 3572 bytes, which is actually smaller than using the direct access method. This is not always useful, because compression increases processing time, depending on the data being compressed.

Further tests were conducted to analyze bandwidth consumption. Medium to large data transfers were realized using web services. This involved accessing an average of 40 megabytes on each test. In [Figure 2.] is illustrated the effect of such data transfers in the absence of other traffic.

Direct database access [Figure 2.(a)] is responsible for an average of 4.5 MB/s bandwidth utilization, while with regular web services [Figure 2.(b)] this increases to an average of 9.9 MB/s, for periods of about ten minutes, during each test. When compression is added [Figure 2.(c)], the required bandwidth drops to an average of 1.6 MB/s, but the processor usage increases from 0.4% to 40%.

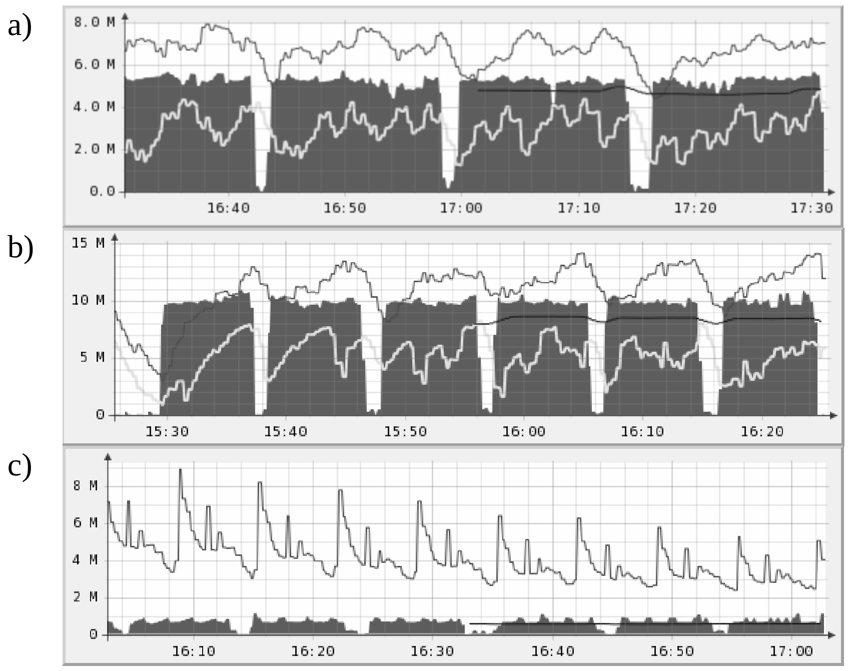

FIGURE 2. Bandwidth consumption when transferring large amounts of data:
(a) Direct database access
(b) Regular web service
(c) Compressed web service.

The last series of tests were conducted in a simulated real network environment. In this case, it has been observed, as expected, a huge increase in HTTP traffic. This led to packets being dropped, especially those related to videoconferencing protocols.

Quality of Service (QoS) techniques were put in place to artificially limit the web service related usage. However, this was possible only by limiting the entire HTTP traffic. Specific messages, related to web services, can not be differentiated from other usage, like browsing the Internet. This poses a great challenge when trying to ensure that data transfer does not interfere with other network activities.

The only identified solution was to use different servers for hosting web services and apply QoS based on host address, rather than content type.

However, this makes it impossible to differentiate between services hosted on the same server. Therefore it is very hard to ensure different bandwidth usage patterns for web services accessed through the same TCP port (like the standard port 80, used for HTTP). Nevertheless, it is possible to further separate services using different ports.

USING CACHING TO IMPROVE WEB SERVICE RESPONSE TIMES

Data sources situated in a distant location must be accessed over the Internet, using much slower connections, compared to local ones. Therefore, even if bursts of several megabytes per second are possible in the local network, such huge requirements can not be easily met over the external connections. Also, because QoS can not be properly used to strictly control web service traffic, all connections to a particular server must be limited, potentially leading to users not being able to access other provided services, during data transfers.

A standard mechanism for accelerating HTTP traffic is provided by network caches. Nevertheless, it is very difficult to use a general cache for web service traffic. This is happening because HTTP requests used with web services must be forwarded each time to the destination server, according to HTTP specifications.

Having in mind the limitations that general caches are faced with, a new application has been built [8]. This is an application controlled cache that presents itself as a web service. Every program using it can control the amount of time a certain entry is stored, thus making it adapt to its own needs.

This new web service is placed in the local network and is used like a buffer between the user application and the remote web services. After data is retrieved from the remote servers, it is placed in the local cache. Therefore, future access to the same data can be handled locally. The lifetime of this copy can be set by the program using it, depending on its usage patterns.

In this case, external data transfers are performed only when data is not available from the locally stored copy. After the initial transfer, future requests are served from the local network, thus reducing external bandwidth requirements and increasing the application’s overall response time.

In [Figure 3.] is presented the effect of caching over the external connection.

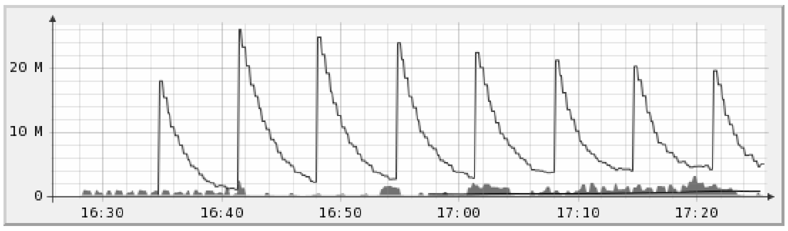

FIGURE 3. Bandwidth usage when using an application controlled caching mechanism [8].

After the first transfer of information, the only traffic traversing the external connection is due to authentication functionality. Actual data is served from local servers, thus conserving the available bandwidth.

CONCLUSIONS

Given the system independent nature of data encapsulated by web services, combined with the ability to use them from almost any modern programming language, new applications are starting to rely on web services for remote data access. However, integrating web services in existing infrastructures raises some rather challenging problems.

After presenting the results of tests, relating to increase in network usage due to web services, several possible solutions were presented in this paper. In order to ensure proper bandwidth requirements for web services, it was suggested to use address and port separation. Then, for reducing the actual data being transferred over low speed connections, it was presented an application controlled caching mechanism.

REFERENCES

1. V.F. Pais and V. Stancalie, "Using Web Services for Remote Data Access and Distributed Applications", *Fusion Engineering and Design*, Volume 81, Issues 15-17, July 2006, Pages 2013-2017.
2. http://www.w3.org/XML/.
3. http://www.w3.org/TR/wsdl.
4. http://www.w3.org/XML/Schema#dev.
5. P. Calyam, W. Mandrawa, M. Sridharan, A. Khan, P. Schopis, "H.323 Beacon: An H.323 application related end-to-end performance troubleshooting tool", *Proceedings of ACM SIGCOMM Network Troubleshooting Workshop (NetTs 2004)*, Portland, October 2004.
6. http://www.postgresql.org/.
7. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt.
8. V.F. Pais and V. Stancalie, "Caching Web Service for TICF Project", *Fusion Engineering and Design*, 2008, in press, doi:10.1016/j.fusengdes.2007.11.008.