Performance Comparison of Web Services under Simulated and Actual Hosted Environments

Sandesh Tripathi
Integral University
Kursi Road, Lucknow
India

S Q Abbas
Integral University
Kursi Road, Lucknow
India

ABSTRACT
Web Services are modular applications that are published, advertised, discovered and invoked across a network, i.e, an Intranet or the Internet. It is based on the software services model, in which these may participate as individual or as a component of other services and applications. This research follows a performance testing approach for Web Services under simulated and actual hosted environment. The study compares the performance parameters –response time, throughput for web services, which helps the developer in early development life cycle of web services. Such study helps in tuning the applications before putting it before the world. Our measurements suggest that from modeling perspective web services can be simulated first and tested for various performance metrics, which give results close to the original one.

Keywords
Web Services Performance, Performance Modeling, Throughput, Response Time.

1. INTRODUCTION
Web Services are the software technology that uses XML to share data. It is used to expose methods over web through SOAP (Simple Object Access Protocol), described by WSDL (Web Services Description Language), registered in UDDI (Universal Description, Discovery and Integration). WSDL enables language independent description of types, messages, port types and port used in a web service. The server makes these methods available. These methods when called will perform some action and/or return some data. The web services standard spells out in great detail how a client machine can invoke a web service method from a server. The significant advantage of using web services over the previous interoperability attempts, such as CORBA (Common Object Request Broker Architecture), is that it utilizes open standards based on Web/Internet ubiquitous technologies such as XML, HTTP, and SMTP. Herein the SOAP payload is widely transported using mainly HTTP protocol. [18]

Many applications in the areas of finance, distributed computing, e-commerce and GIS have been exposed using web services, but performance issues of Web Services may limit their applicability in some situations. Compared to RMI, Web Services put few restrictions, such as no support for object references, absence of distributed garbage collection etc. The eXtensible Markup Language (XML) produces human readable text and is emerging as the standard for data interoperability among WSs and cooperative applications that exchange data. XML documents consist of elements, tags, attributes, etc. and satisfy precise grammatical rules [2]. Due to the use of XML based protocol (SOAP) performance will be a concern.

This study performs test on web services under simulated environment and hosted environments.

This paper is structured as: Section 2 details the related work, Section 3 describes the performance bottlenecks, Section 4 describes the performance evaluation, measurement and simulated environment, Section 5 explains the result and outcomes.

2. RELATED WORK
Performance in web services is an open problem. In order to study the performance of Web Services many studies have been done.

In paper [3] Osama Hamed and Nidal Kafri presented a performance testing approach that aims to utilize load testing tools to give ideas about performance issues early in development life cycle for applications implemented using Java EE and .net Platform.

Eduard and Zeigler [6] identified and presented relevant differences between network management and simulation for testing network mechanisms. But such a study was still a need concerning the performance of Web Services.

Martiz B. juric etc. [10] compared different approaches for distributed Java applications, which communicate through fire wall and proxy secured networks. The study concludes that RMI performs better than web services.

Govindaraju etc. [12] compared the performance of gSOAP, Axis C++, Axis Java, .net and XSOAP/XSUL toolkits. This paper focused on SOAP performance on scientific data. It raised the features of SOAP that affect Web Services performance.

Ng. etc [17] studied the latency throughput and serialization and deserialization overheads of different messages exploring the encoding styles supported by each evaluated toolkit.

Ana C. C. Machado. Carlos A. G. Ferraz.[18] presented guidelines of two mostly used Java Web Services toolkits- Apache Axis and Sun Java web Services Developer pack (JWSDP),they also gave recommendations on the use of SOAP features to guarantee interoperability between toolkits.

Lisa Wells, etc. [9] presented a general framework for modeling distributed computing environments for performance analysis by means of timed hierarchical coloured petri nets.

Vast numbers of performance studies have been done but there was a need to compare the results under simulated and hosted environments for web services.

3. PERFORMANCE BOTTLENECKS

Serialization is an integral part of both web services and network centric data base applications. The process of converting data from one format to another is known as serialization. With a network connection, the receiving process consumes arriving messages, re-assembles the transmitted objects and processes them. Once the data has been serialized, at a later point of time it has to be deserialized Web Services spend considerably more time for XML serialization and deserialization, which is used for creating SOAP messages (com.sun.xml.rpc package).[10]

Two common encoding mechanisms are XML and Binary. XML produces human readable text and is employed when interoperability with other web services is essential. Binary produced streams are compact to parse, but not human readable.

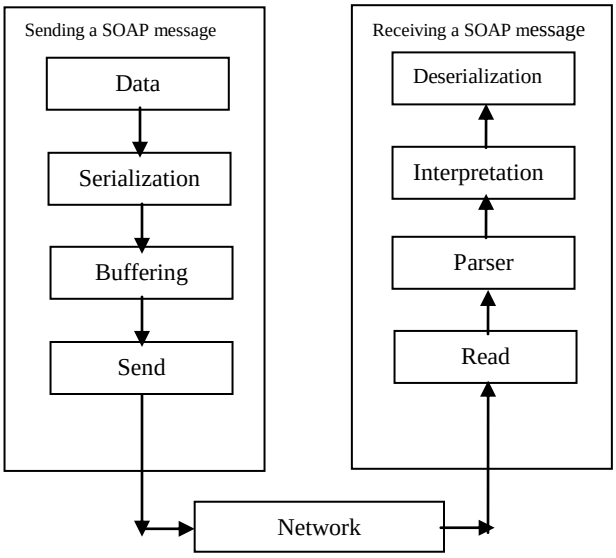

Figure 1: Stages to send and receive a SOAP message

3.1 Message Size of XML Messages, Serialization, Deserialization and network level over heads

It is suggested that [11], when the message size is large, i.e. in comparable size of RAM, XML encoding diminishes the Response time, because it inflates the message size, causing the operating system to write either a part or the entire message to disk. XML encoder results in message size that are at times five times larger than their binary representation. In some [18] cases it is even 4 to 10 times bigger than its binary version. This expansion in message size have greater impact on communication and on the total RTT , in this way it affects the throughput requiring more processing time.

The serialization of Java Objects in XML consumes 10 times more memory than the binary serialization and that the cost associated with communication and copy of the data are lesser than the costs of serialization and deserialization. It has been identified that 50% of the total time are spent in the codification of the SOAP messages in XML before of the transmission for the server and in creation of the HTTP connection. Whereas the .Net Frame work provides a number of serialization options. The one that is pertinent to Web Services is XMLserialization, which involves serializing in memory data to an XML format. With Web Services this XML formatted version of data is sent from the client consuming the web service, to Web Service, or vice-a-versa. The nice thing about the .NET framework's serialization capabilities is that a developer, have to do a very little.

The overhead introduced by the HTTP protocol is also considerable. [16] The specification demands that the client establishes a new connection before each request and that the server to close connection after sending the response. The responses start by indicating which version of HTTP server is running, followed by a series of optional object headers; the most important among them is 'Content-type', which indicates the type of object being returned and the 'Content-Length', which indicates the length. The header is terminated by an empty line. The server now sends any requested data. After the data have been sent, the server drops the connection. HTTP it seems spends more time waiting than it does in transferring data.

3.2 Connection Establishment, Data Transfer: Windows and Slow Start

TCP establishes connections via a three way hand shake. The client sends a connection request, the server responds, and the client acknowledges the response. The client can send data along with the acknowledgement. Since the client must wait for the server to send its connection response, this procedure sets a lower bound on the time of RTTs.

Instead of having to wait for each packet to be acknowledged, TCP allows the sender to send out new segments even though it may not have received acknowledgement for previous segments. To prevent the sender from overflowing the receiver buffers, in each segment the receiver tells the sender how much data it is prepared to accept without acknowledgement. This value of window size, tells the sender the maximum amount of unacknowledged data that the receiver is prepared to let it have outstanding, the receiver cannot know how much data the connecting networks are prepared to carry. If the network is quite congested, sending a full window worth of data will even cause worse congestion. TCP determines the best rate to use through a process called Slow Start. With Slow Start, the sender maintains and calculates a second window of unacknowledged segments known as Congestion Window. When a connection first starts up, each sender is only allowed to have a single unacknowledged segment in transit. Every time a segment is acknowledged without a loss, the congestion window is opened; every time a segment is lost and timed out, the window is closed.

This approach is ideal for normal conditions; these connections tend to last a relatively long time, and the effect of slow start is negligible. For short lived connections, the effect of slow start is devastating.

4. PERFORMANCE EVALUATION

The model used for web traffic generation is called SURGE, published in paper [8]. In order to get a constant load for each scenario the numbers of users are fixed. SURGE derives probability distributions for web traffic properties. For the comparison, the surge traffic generator is implemented for both simulation and measurement. For simulation the generator is tightly integrated in the ns network simulator, for the measurement it is implemented as a traffic generator application. This generation application (GA) is divided into a WS server

application which listens for connection requests from clients WSs. A client WS (GA) sends requests according to the SURGE random variable. Network access is accomplished by means of socket interface. To manage parallel connections a thread is spawned for every connection. The client and server part contains logic for efficiently managing load generating hundred users per host.

4.1 The measurement and simulated environment

The hardware platform consists of two PCs. One is server and the other is client. The client and server was connected using LINKSYS Ethernet switch. The server is configured with a Microsoft Windows 2000 Server, Visual studio .net. The server implements a Web Service which accepts query from the client and returns back the client.The test bed used for measurement consists of X86 architecture node with Free BSD 4.4 installed and is used either as server, client or router. All host and router PCs are connected with 100 Mbps full-duplex links in order to avoid collisions. The bottleneck link between routers is emulated using a software link emulator known as dummynet, which is a part of Free BSD kernel.

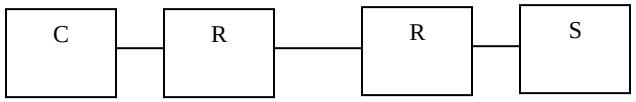

Figure 2: Test Topology

The server part of the traffic generator is installed on node S, machine while the client part is installed on C machine. Using this topology the network environment is tested with low load (20, 50 users) to medium load users (100, 400 users).

The Simulation part of this comparison is performed with the network simulator ns. The effort has been done to make the simulation as close to measurement as possible. The bottleneck emulation is done by dummynet, which is used in the test bed, has been added to ns. The ability of dummynet emulating the bandwidth depends on the system tick interrupt, which is set to a period of 10ms on PC type h/w. All the dispatching of the packet has been done when the tick handler function is called. This may result of a packet burst up to 25packets per 10ms, creating a *dummynet* bottleneck of 10 Mbps and 4000 packets.

5 RESULTS

5.1 Arrival Rate of Packets

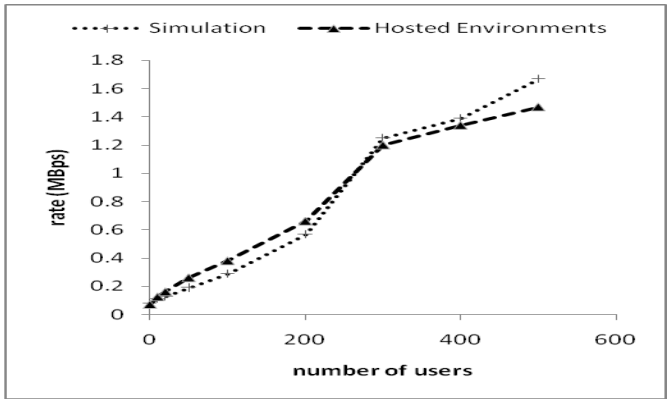

Figure 3: Arrival rate of Packets

The arrival rate is calculated by culminating the size of packets arriving during a time at the input of the bottleneck queue. The average has been obtained by calculating mean for total time intervals. The value gives us rough information about the bottleneck load for web services.

The rates in measured and simulated environments are very similar but have a reproducible difference. While the medium load scenarios ranging from 80 to 300 users expose a higher rate for measured environment. This situation takes a reverse turn for higher load scenarios. The arrival rate is a metric for the load on the bottleneck link, it can be concluded that both scenarios impose an equal demand on the bottleneck.

5.2 Response Time and Throughput

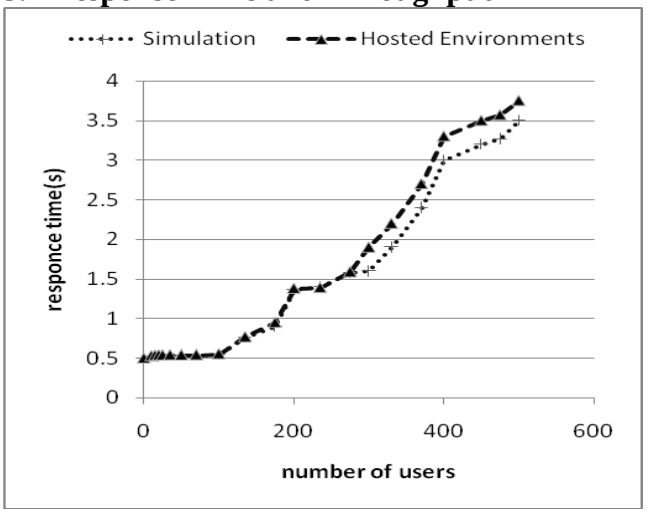

Figure 4: Response Time

The response time for web service is the time delay observed from when a client invokes a remote web service until it receives the last byte of response produced by the service. After looking at the results from data analysis point of view it seems that there is no difference in the result, from network point of view. The response time in hosted environment follows the simulation behavior up to 300 users. After that response time for hosted environments crosses the simulation.

The number of active flows increases in dependence on the number of users. In the simulation, traffic generation model has a weak dependence on the capacity bottleneck for low load scenarios and the traffic amount is basically only dependent on the number of virtual users. The higher queue size in measurement in the high load scenarios can be explained by a higher variance of traffic arriving at the bottleneck. Due to higher mean queue size in hosted environments, a higher proportion of packets are dropped, causing the TCP sender to reduce the sending rate. This is why a considerable higher response time in hosted environments is observed.

The throughput is the number of active requests processed by the environments given per unit time. The trend for hosted environment and simulation continues as the same. Hosted environments show a comparable throughput for medium size packets, and falls below simulation for large size packets. But not considering the bottlenecks of the network the results can be considered equal.

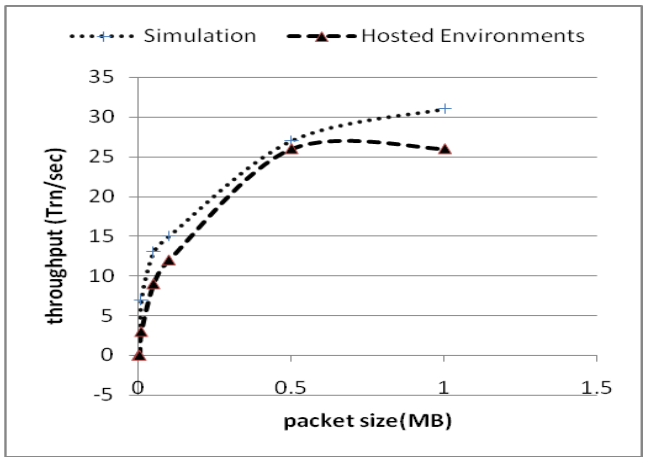

Figure 5: Throughput

6. CONCLUSION

Web Services are chosen in terms of in terms of interoperability, simplicity, flexibility and reuse of services. Efforts are going on to reduce the performance limitations of Web services. This paper summarizes the performance bottlenecks for Web Services.

In order to get true feeling before implementation, it is useful if some trends are available. This paper presents the comparison of few Web Services performance parameters in the simulated and hosted environments. Since simulation and actual hosted environments show similar results, web services can be first simulated and tested before actually putting them on the web.

7. REFERENCES

[1] Menasce, D.A., Almeida, V.A.F.Dowdy, L.W (2004). Performance by Design: Computer Capacity Planning by Example. Prentice Hall, 2004

[2] Shahram Ghandeharizadeh, Christos Papadopoulos, Parikshit Pol, Runfang Zhou. NAM:A Network Adaptable Middleware to Enhance Response Time of Web Services. In proceedings of the IEEE/ACM International symposium on Modelling, Analysis and Simulation of Computer Telecommunications Systems.

[3] Osama Hamad, Nedal Kafri. Performance Prediction of web based Application Architecture Case Study: .Net vs JavaEE. International journal of Web Applications, Vol1,Number3,146-155

[4] Aniruddha S. Gokhle, Douglous C.Schmidt, Measuring and optimizing CORBA latency and Scalability over high Speed Networks.IEEE Transactions on Computers, Vol.47,No.4,April 1998

[5] V. Almeida, J.Almeida, and C.Murta.Performance analysis of a WWW Server. Technical Report 1996-018, Computer Science Department, Boston University and UFMG. Aug. 1996

[6] Eduard Hasenleithner Thomas Ziegler. Comparision of Simulation and measurement using state of art web traffic models. In proceedings of the Eigth IEEE International Symposium on Computers and Communication(ISCC'03)

[7] R.van der Mei, R. Hariharan, and P. Reeseer. Web Server performance modeling. Telecommunication systems, 16(3-4):316-378, 2001

[8] P.Badford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In proceedings of the ACM SIGMETRICS conference, Madison, WI, July 1998

[9] Lisa Wells, Soren Christensen, Lars M. Cristensen, and Kjed H. Mortensen. Simulation Based Performance Analysis of Web Servers. In Proceedings of the 9th International Workshop on Petri Nets and Performance Models (PNPM'01), 2001

[10] Matjaz B. Bostjan Kezmah, Marjan Hericko, Ivon Rozman, Ivan Vezocnik. Java RMI, RMI Tunneling and Web Services Comparision and Performance Analysis. ACM SIGPLAN ,58-65, Vol.39(5), May 2004

[11] M. Cai, S. Ghandehrizadeh, R. Schmidt, and S. Song. A Comparison of Alternative Encoding Mechanisms for Web Services. In proceedings of DEXA Conference, August 2002

[12] Govindaraju, M. Slominiski, A. Chiu, K. Liu, P.etc. Characterizing the Performance of SOAP toolkits. In proceedings of the 5th IEEE/ACM International Workshop on Grid Computing. Pittsburg, U SA, 2004,365-372

[13] Nathan Meyers, Java Programming on Linux, Macmillan Publishing, 1999

[14] M.Artit and C.Williamson. Internet Web Servers: Work Load Characterization and Performance Implications,IEEE Transactions on Networking,5(5):661-645,1997

[15] Davis, D. and Parashar, M. Latency Performance of SOAP Implementations. In the proceedings of the 2nd IEEE/ACM International symposium on Cluster Computing and Grid Computing(CCGRID).Berlin, Germany,2002,785-790

[16] Simon E. Spero. Analysis of HTTP performance problems

[17] Ng,A. Chen, S. and Greenfield, P. Evaluation of Contemporary Commercial SOAP. In proceedings of the 5th Australian Workshop on Software and System Architectures (AWSA), Melbourne, Australia, 2003, 64-71

[18] Ana C. C. Machado. Carlos A. G. Ferraz. Guidelines for Performance Evaluation of Web Services. WebMedia'05, December 5-7, 2005, Pocos de Caldas, MG, Brazil

[19] F.Cohen, "Discover SOAP encoding's impact on Web Service Performance", http://www.128.ibm.com/ developerworks/webservices/library/ws-soapenc/

[20] T.Bell, J.Cleary, and I.Witten. Text Compression. Chapter 1, Prentice Hall, Englewood Cliffs, NJ,1990.