International Journal of Database Theory and Application
Vol.8, No.3 (2015), pp.297-312
http://dx.doi.org/10.14257/ijdta.2015.8.3.26

A Comparative Investigation on Implementation of RESTful versus SOAP based Web Services

Abhijit Bora and Tulshi Bezboruah

Department of Electronics and Communication Technology, Gauhati University, Guwahati-781014 Assam, India
Tel: +91-361-2671262(O); Fax: +91-361-2700311(O);
abhijit.bora0099@gmail.com / zbt_gu@yahoo.co.in

Abstract

Investigations on web service performance metric based on RESTful architecture against conventional SOAP based architecture has importance in perspective of developers as well as for end users. As such we have developed and hosted two web services, one based on SOAP and the other based on RESTful architecture. Both the services are based on JAVA technology implemented with apache tomcat web server and MySQL as backend database server. A comparative evaluation of both the web services is carried out to study its scalability, efficiency and feasibility. Load and stress testing tool Mercury Load Runner is used to deploy both the services for testing the architecture. The statistical analysis on recorded performance metrics is carried out to study the effectiveness of the services. This paper presents in details the comparative analysis of the experimental results on performance aspects of the services.

Keywords: *RESTful, SOAP, Web Service, Java, RDBMS*

1. Introduction

The Web Service (WS) is one of the popular hypes in the software industry today. It provides interoperability and unlimited connection with new business opportunities. The software interoperability concept is not new. There have been a number of implementations that give solutions for this concept. The Remote Procedure Call (RPC), Open System Interconnection (OSI), Common Object Request Broker Architecture (CORBA), Remote Method Invocation (RMI) is among some of them.

The WS is a software application that can be accessed over the network [1]. Business to Business (B2B) integration by aggregating WSs enhancedit to a hierarchical WS communications [2]. Every WS may play the role of a broker, and a service provider [3] that can be called by a client application.

A WS provides flexibility for establishing communication between geographically separated systems or devices over the internet. Millions of WS are published across the internet which can be used, according to the requirements of the consumers. These services may be available as Web Service Description Language (WSDL) files or sometimes the services might be available directly. The growing popularity of WS can be ascribed to a movement towards Service-Oriented Architecture (SOA).

While considering WS for implementation, many factors are to be analyzed. The involved communities and industrial parties are concerned about the performance aspects. When creating WS there was a trade off where performance was sacrificed for simplicity and flexibility. The performance part will have a great impact as it directly reflects the costly investments in new hardware. In this paper we focus on performance aspects of SOAP based and RESTful WS, as these two WSs are popularly used as online service between consumers and service providers.

ISSN: 2005-4270 IJDTA

1.1. RESTful WS

Representational State Transfer (REST) [4] is software application architecture where functionality and data resources are accessed using Uniform Resource Identifiers (URIs). It is client-server architecture, used as a stateless communication protocol, such as Hypertext Transfer Protocol (HTTP) to exchange resources through standardized interface. These principles make REST applications to be light weighted and gain high performance. The WS built upon the REST architecture is called RESTful WS. They use the HTTP methods for functional operations with available resources. Systems that follow the REST principles are often called ‘‘RESTful’’ [5]. The key principles of REST architecture include the following notions:

(a) Application state and functionality are abstracted into resources. Any information, which is offered by the system and can be named, is possible to be represented by a ‘‘resource’’. Any concept that needs to be addressed, referenced and accessed must fit within the definition of a resource [6].

(b) Resources must be uniquely identified and addressable using a universal syntax, such as a Universal Resource Identifier (URI) used in HTTP [6].

(c) A uniform interface is shared by all resources for the transfer of state between client and the server. The set of operations, as well as supported content types, need to be well defined. At the same time, code on demand (such as JavaScript) could be optionally supported [6].

(d) The communication protocol between the resource data provider and consumer has to be: client-server, stateless, layered and cache enable.

When the REST architectural principles are applied, as a whole, they provide enhanced scalability, generality of interfaces, independent deployment, reduced interaction latency and they can encapsulate legacy systems . With the advantages and characters of the REST, REST WS are broadly applied in system integration in most of the research fields [6].

As resources are marked with global URIs, they are accessible once they are exposed on the Web rather than a separate resource discovery and location mechanism [10].

The requests and responses for RESTful WSs are typically HTTP messages that are far less in size compared to SOAP messages. Since in REST architecture a resource can be directly identified by its URI, therefore extensive SOAP parsing can be avoided that is required for invoking a service [13].

Each request includes all the necessary information for the servers to understand, so each transaction is independent and unrelated to previous ones. Servers do not need to keep states between requests [14].

1.2. SOAP- based WS

The main platforms of Simple Object Access Protocol (SOAP) based WS are SOAP and WSDL. The SOAP is an Extensible Markup Language (XML) based protocol that allows exchanging information using HTTP. This protocol helps in accessing a WS and provides the flexibility in establishing communication between two software, even they are running on different operating system (OS) with different techniques. It has specifications for stateful implementation. It uses XML for its message format, and relies on HTTP and Simple Mail Transfer Protocol (SMTP), for transmission and message negotiation. SOAP provides messaging framework upon which WS can be built [7].

The WSDL is based on XML and is used to describe and locate the WS. The XML document describes its service and the method names to be invoked. A WSDL document uses a container for data type definitions for the WS such as <types>. A typed definition of the data being communicated is specified through <message> tag. A set of operations supported by endpoints is specified by <portType> tag; A WSDL document can also manage service element to group together several WS definition in one single WSDL

document [8].
SOAP forms the foundation layer of WS protocol stack thereby enhances interoperability with applications running on different operating systems and programming languages. This protocol consists of an envelope, which defines what to be included in the message and how this massage should be processed; a set of encoding rules, and a convention for representing procedure calls and responses [7].
However, it is time-consuming to serialize and de-serialize native languages into SOAP messages. Furthermore, the WS protocol stack is also complex so that only programmers can understand how to deploy a service [9, 11].
Most of the information in the SOAP and WSDL is redundant and meaningless. It increases the network communication volume and server-side payload and it is difficult to support the proxy and cache servers, because clients cannot identify the useful information straightforwardly from the URI and HTTP [9, 10].
Figures 1(a)-(b) show the different steps of routing for request and response of a sample SOAP based WS compared to RESTful WS.

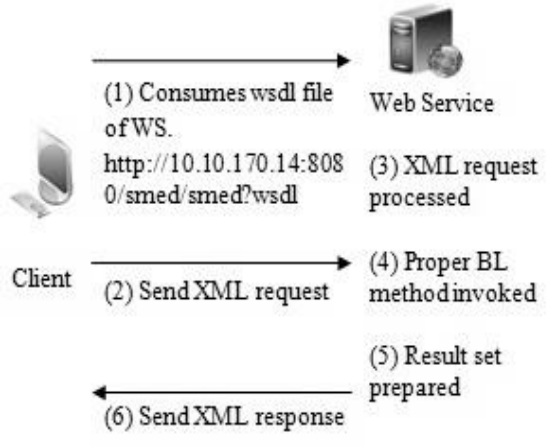

Figure 1(a). SOAP based WS Request Response Routing

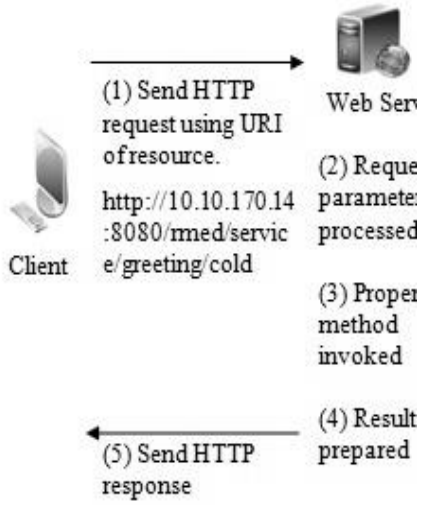

Figure 1(b).RESTful WS Request Response Routing

1.3. Related Work

In the year 2005, M. B. Juric, I. Rozman, B. Brumen, M. Hericko, M. Colnaric [15] presented a study on functional and performance related differences between WS and RMI with WS-security variants. They conducted the test using two identical computers with operating system Whitebox Enterprise Linux 3.0 and Windows XP Professional SP2. The hardware configuration was Intel Pentium 4 processors 2.4 GHz, 512 RAM with Java Web Services Developer Pack version 1.4, Java 2 Platform Standard Edition and Apache Tomcat 5.0 as web server.
In the year 2006, A. E. Saddik [16] presented a methodology for testing the scalability and performance of a specific SOAP based WS application and analyzed the results of the testing. He conducted the test by deploying the service with hardware specification having Intel Pentium 4 Central Processing Unit (CPU) 2.20 GHz, 512 RAM, 80 GB HD, Sun ONE application Server 7, MySQL Server 3.23, 100Mbps switch and Digital Subscriber Line (DSL) modem. He monitored number of successful response, number of error responses, total number of session and percentage of error response for a load level of 10, 100, 500 and 1000 agents.

In the year 2009, J.Meng, S. Mei, Z. Yan [9] presented an analysis about traditional WS and RESTful WS and designed a testing scheme to test and analyze the performance. A traditional WS is developed using Microsoft Visual Studio.Net 2003 on IIS5.1 in C#, while the other using MyEclipse on Apache Tomcat 5.0 in JAVA1.6. RESTful WS is implemented on Rails2.2 in Ruby using the IDE of RadRails. They monitored the average response time of the WS and the size of the response packet of the server. They process the same business logic and share the same data source.
In the year 2011, BipinUpadhyaya, Ying Zou, Hua Xiao, Joanna Ng, Alex Lau [17] presented an approach to migrate SOAP-based services to RESTful services and measured the effectiveness of the approach through a case study.
In the year 2012, KamalEldin Mohamed, DumindaWijesekera[18] presented an evaluation and a comparative analysis on average response time for testing RESTful and SOAP-based WS on mobile devices.
In the year 2013, Ricardo Ramos de Oliveira, Robson Vinícius Vieira Sanchez, JúlioCezar Estrella and RenataPontin de Mattos Fortes and ValérioBrusamolin [19] described an experiment to compare RESTful and SOAP-WSDL WS in terms of specific modifiability sub-characteristics and time spent on WS maintenance. Descriptive statistical analysis was used to evaluate the maintainability of the services in server and client side.
In the year 2013,P.Markey, G. Clynch [20], presented results of performance analysis that was conducted for SOAP and RESTful approaches. The performance metric measured was network weight i.e. the amount of network traffic that resulted from an interaction between a client and the WS. The WS were implemented using Window Communication Foundation (WCF) and the Web Application Programming Interface (API) for the .Net platform in C#, and Internet Information Services 7 (IIS7). The server is also running an instance of SQL Server 2008 R2.

2. Description of Objective and the Methodology of Investigations

To compare the performance aspects of SOAP based WS with its counterpart RESTful WS and to find out the factors that impact the performances is the main objective of the investigations. To achieve the objective, we have developed, implemented and tested two prototype WSs, one based on SOAP and the other based on RESTful architecture considering pharmacological data [21] and analyzed metrics like performance, scalability, load and stability of the system. MySQL database engine, apache tomcat web server and Java programming language was used to develop and implement both the WS.The dataset for the service is 10000. The WS has been deployed on Mercury LoadRunner for performance and load testing. The data arrangement and referential integrity in between diseases and clinical remarks is prepared. The architecture and algorithms are developed for both the WSs. The testing is performed up to 1500 virtual users and responses are recorded accordingly. Statistical evaluation has been performed on performance metrics of WS to study different aspects of the service.

3. Software and Hardware Environment

The open source Java language is the general choice for developing WS. Its strong security mechanism, concurrency control and wide spread deployment in both client and servers makes it relatively easy to create WS [22, 23, 24]. The WS application can be developed using Java programming language. The software specifications used in the work are: (a) Integrated Development Environment (IDE) platform: NetBeans version 7.0, (b) web browser: Google Chrome, (c) web server: Apache Tomcat version 7, (d) the database engine: MySQL version 5.0. The client and the WS have been hosted on server with 64-bit Windows Server 2008 R2 Standard operating system (OS). The hardware

specifications are: Intel® Xeon® CPU E5620 @ 2.40 GHz; 8 GB RAM and 600 GB Hard disk.

3.1. The Architecture

The architecture for the SOAP based WS is presented and discussed elsewhere [25]. The architecture for the RESTful WS is shown in Figure 2. This architecture represents the following services.

3.1.1. The Client Spplication of the Service: The client as a consumer application of the WS contains the user interface (UI) for capturing the end user data. It captures and sends the data to RESTful WS.

3.1.2. The RESTful WS: This service contains the necessary BL operations related to the data processing. The RESTful WS manages: (i) the data mapping, (ii) result set generation, (iii) insertion and (iv) fetching of data from the database. It captures the required parameter from the client service and executes the SQL statement for database operation. The RESTful WS holds the database queries for performing necessary operation.

4. Design Aspects of the Service

The prototype research WS is based on Java technique suitable for rural and urban clinical health services. We call it prototype research medical web service (MedWS). The clinical details of diseases have been taken into account such as: (a) the medicine name, (b) manufacturing company name, (c) the component category of medicine, and (d) the tablet, syrup, injection, and lotion package information *etc*.

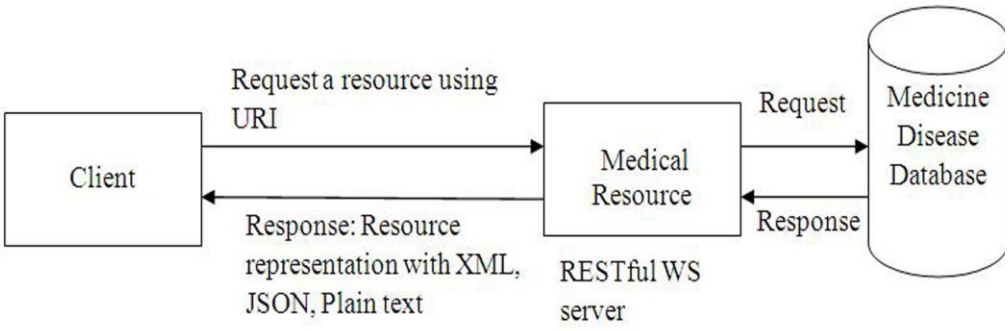

Figure 2. Architecture of the Proposed Medical WS based on RESTful Architecture

We merge them all to prepare a clinical advice for hosting MedWS. The service follows a pharmacological book published in India as sample data [21]. When users access the URL of the client application, the HTML form will open. This interface allows entering a particular disease name in the textbox. The response page will appear by fetching the records available in the database.

4.1 The Algorithm

We have developed the algorithms for both the services. The algorithm for developing the RESTful WS and SOAP based WS is given below:

Algorithm 1: Algorithm for Developing the RESTful WS

Step	Instruction
1	Begin
2	Establish root resource class
3	Capture parameter received from client
4	Establish a method to process GET request from client
5	Specify the MIME media types of representations
6	Establish database connectivity using JAVA bean
7	Execute a Structured Query Language (SQL) SELECT statement using the passed parameter
8	Get the resultset
9	If size of resultset greater than 0
10	Go to step 12
11	Else go to step 15
12	Arrange the resultset
13	Assign the resultset to a response object
14	If Success go to step 16
15	Create an empty response object
16	Return the response to the client
17	End

Algorithm 2: Algorithm for Developing the SOAP based WS

Step	Instructions
1	Begin
2	Create a WS operation
3	If success go to step 5
4	Else go to step 2
5	Capture parameter received from client
6	Establish database connectivity using JAVA bean
7	Pass parameter to specific method
8	Execute a SQL SELECT statement using the parameter
9	Get the resultset
10	If size of resultset greater than 0
11	Go to step 13
12	Else go to step 16
13	Arrange the resultset
14	Assign the resultset to a response object
15	If Success go to step 17
16	Create an empty response object
17	Return the response object
18	End

5. Testing of the Services

Both the servicesare deployed on Mercury Load Runner version 8.1 for testing. It helps to predict the systems' performance and behaviors. It stresses the service by creating virtual users, recording the systems' performance metrics and then analyzes it [26]. During the experiments, we set approximately 30sec as users think time to perform the transaction and assigned 5min as steady-state period for all the tests. The stress level is gradually varied to saturate the server. We follow the various test steps presented elsewhere [23].

The SOAP based WS invocation test case is given elsewhere [25]. The test case for RESTful WS is shown in Table 1 below.

5.1 Testing Benchmark

The settings of various parameters during testing procedure includes: (i) the Virtual User (VU)-stress margin, (ii) the user think time, and (iii) the network speed. The VU-stress margin defines the number of accessing users of the service, the user think time specifies the time that an end user will take in thinking before requesting the service, and the network speed states the network bandwidth (BW) that the VU will use.

5.1.1 Scalability Testing: It describes a WS's capability to serve clients under varying level of load [27]. To measure scalability, we can run a test script for sending request to MedWS and can measure their response times. It measures when valid test clients are completed correctly.

5.1.2 Performance Testing: It is a twin to scalability that evaluates the WS's ability to accurately deliver functions [16]. During this test, we measured the ability of the services on how well it performs under different load conditions with variable amount of stress level [28].

5.2 Test Responses of the Service

The recorded performance attribute of our test include: (a) the hits/sec, (b) the throughput, (c) the response time and (d) the number of VU that performed successfultransaction.

Table1.Test Case for RESTful WS Invocation

Step	Actions to be performed	Expected outcome
1	Open browser and access the RESTful WS URLhttp://server1/spr0Client/index.jsp	Client's home page containing a HTML form is displayed
2	Enter disease name such as "Cold" and click "Submit" button	Send the data to RESTful WS resource for necessary BL invocation. Prepare the clinical result set.
3	Response page is generated	The web page http://server1/spr0Client /result.jspis generated. It contains a tabular format of clinical information.

5.3 Analysis of Experimental Data and Evaluation

The testing has been carried out for 100, 150, 200, 250, 350, 500, 600, 700, 800, 1000, 1200, 1500 VUs with 1 Gbps BW. The load with ramp up schedule is set as 1 VU entering into the script after every 15s. The test duration of 5min is set after ramped up of all VUs to record steady-state measurement. The VUs ramp down simultaneously after the completion of the steady-state. We deploy both the WS on Mercury Load Runner with 1000, 1200 and 1500 VUs to observe the connection refusal. The experimental results are shown in Table 2.The sample responses of performance test for 350 VUs are shown in Figures 3-4.

Figure3 depicts the response time of SOAP based WS. It acquired maximum at 311 VUs and then the response time falls down gradually. The average response time of 350 VUs is observed to be 11.406 with a maximum of 12.527.
Figure4 depicts the response time of RESTful based WS. It acquired maximum at 326 VUs and then the response time falls down gradually. The average response time of 350 VUs is observed to be 2.556 with a maximum of 14.431. In both cases, it is seen that response time is proportional to number of VUs.

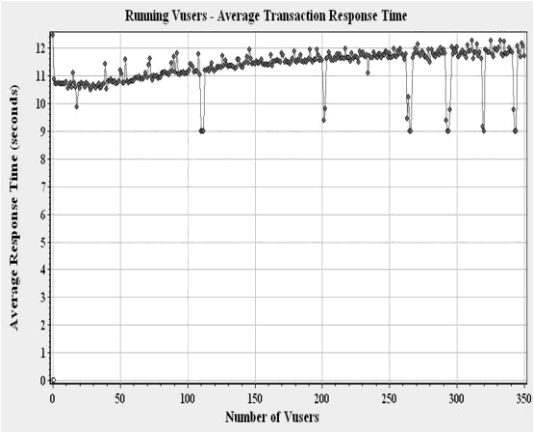

Figure3. SOAP based WS Response Time Against 350 Virtual Users

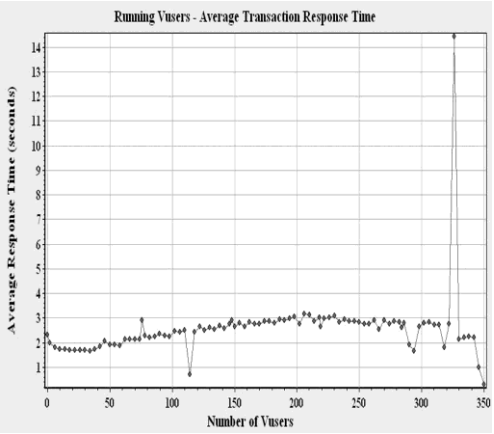

Figure 4.RESTful WSResponse Time Against 350 Virtual Users

6. Statistical Analysis and Evaluation

The statistical analysis on recorded metrics of 50 users is presented here. A sample of 30 repetitive tests is taken for statistical evaluation. The performance metrics are categorized into 6 different classes as per their frequency range. The frequency intervals of response time for the SOAP based and RESTful WS are shown in Table 3 and 4 respectively.

Table2.Experimental Results for RESTful and SOAP based WS

No. virtual user accessing the WS	Recorded parameter	RESTful WS		SOAP based WS	
		Average	Connection refusal in %	Average	Connection refusal in %
100	Response time (s) Throughput (bytes/s) Hits/s	2.123 2226 0.95	0	10.020 2489.270 0.681	0
150	Response time (s) Throughput (bytes/s) Hits/s	2.258 2912.070 1.272	0	10.412 4726.324 0.953	0
200	Response time (s) Throughput (bytes/s) Hits/s	2.502 3840 1.645	0	10.424 6934.176 1.181	0
250	Response time (s) Throughput (bytes/s) Hits/s	2.794 4659.759 1.939	0	11.564 8334.323 1.455	0
350	Response time (s) Throughput (bytes/s)	2.296 14329.717	0	11.709 11015.608	0

	Hits/s	4.724	1.966
500	Response time (s) Throughput (bytes/s) Hits/s	1.847 0 14809.701 3.682	11.991 0 14135.344 2.638
600	Response time (s) Throughput (bytes/s) Hits/s	2.930 0 10438.245 4.211	10.358 0 9425.305 3.146
700	Response time (s) Throughput (bytes/s) Hits/s	2.250 0 11919.066 4.880	11.816 0 16043.586 3.649
800	Response time (s) Throughput (bytes/s) Hits/s	2.047 0 14034 5.669	11.776 0 16834.457 11.776
1000	Response time (s) Throughput (bytes/s) Hits/s	1.778 0 16071.575 6.673	9.935 1 9853.486 4.990
1200	Response time (s) Throughput (bytes/s) Hits/s	1.289 0 18713.023 8.02	10.734 14 11885.978 5.229
1500	Response time (s) Throughput (bytes/s) Hits/s	1.268 0 22610.054 9.811	11.019 61 9.49.034 2.969

Table 3. SOAP based WS

Bin	Frequency
10.424	1
10.512	1
10.6016	3
10.6904	11
10.7792	12
>10.7792	2

Table 4.RESTful WS

Bin	Frequency
1.662	1
1.698	5
1.734	8
1.77	11
1.806	3
>1.806	2

6.1 Distribution of Response Time

The main objective of the present study is to observe the distribution of response time for both the WS. We examine the histogram, quantile plots and normal probability plots of the observed response time of SOAP based WS. The corresponding plots are shown in Figures 5(a)-(c). The histogram, quantile plots and normal probability plots of the observed response time of RESTful WS is shown in Figures 6(a)-(c).

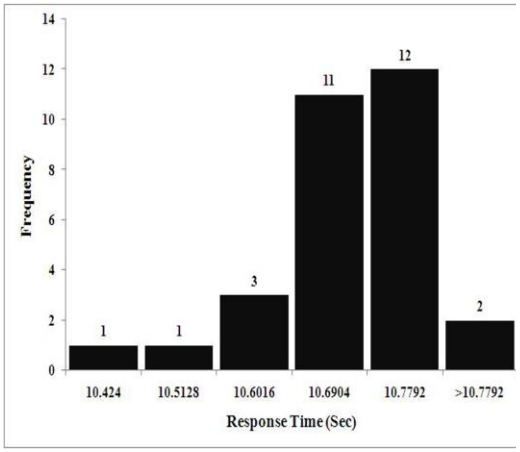

Figure 5(a). Histogram of SOAP based WS Response Time

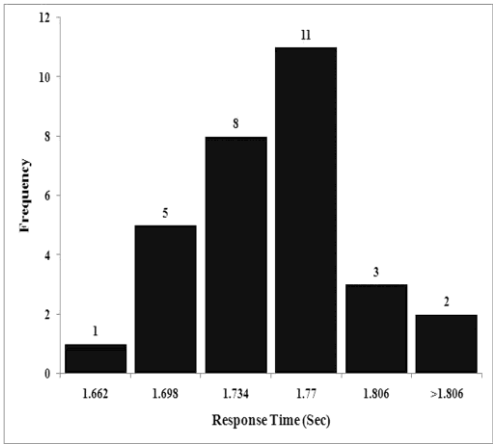

Figure 6(a). Histogram of RESTful WS Response Time

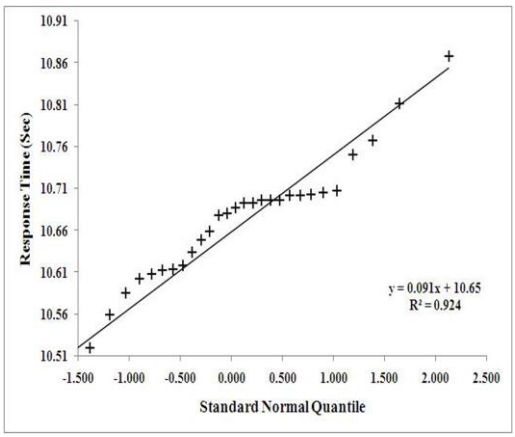

Figure 5(b). Quantile Plot of SOAP based WS Response Time

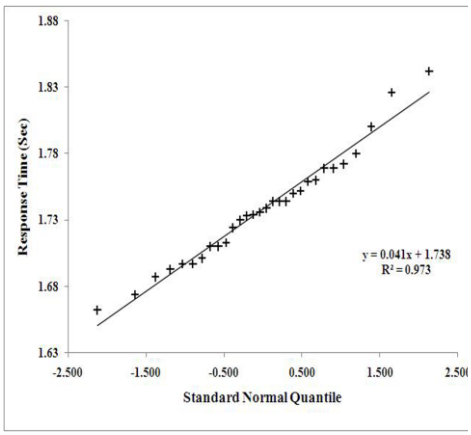

Figure 6(b). Quantile Plot of RESTful WS Response Time

According to the histogram the distribution is normal with slightly left skewed for SOAP based WS and normal for RESTful WS. However, we may find some drawback in histogram, that is, based on the used frequency sizes; it is possible that we may observe different plots. A better technique is to observe a quantile plot. The quantile plot is close to be linear if the distribution of the data is normal in nature [22, 23, 24, 29, 34]. Based on the recorded metrics, the response time attribute of SOAP based and RESTful WS do seems to be distributed normally.

The normal probability plot can be used as a graphical technique to verify the normality of the data samples. If there existed normally distributed data samples, then a linear plot will do appear. Here the most of the data samples are following a straight line. It gives evidence that the distribution is normal one.

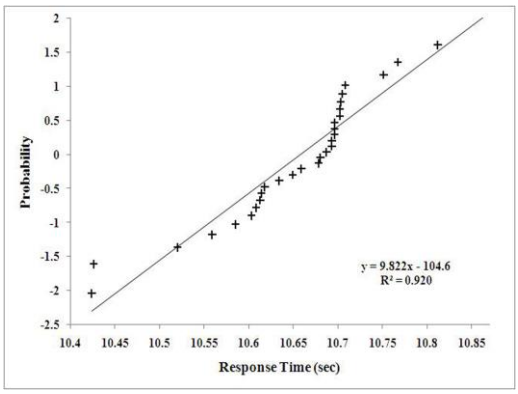

Figure5(c). Normal Probability Plot of SOAP based WS Response Time

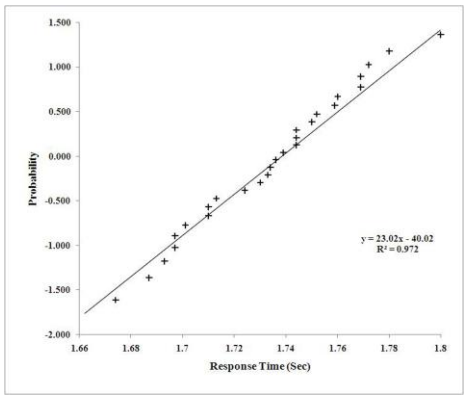

Figure 6(c). Normal Probability Plot of RESTful WS Response Time

6.2 Regression Analysis

The multiple linear regression analysis is carried out using Microsoft Excel to study the combined influence of throughput and hits/sec over response time for both of the services. The regression test is performed with an assumption of null hypothesis (H_0): response time of WS does not depend on hits/sec and throughput. The alternative hypothesis (H_1): response time of WS is dependent on hits/sec and throughput. The results of analysis of variance (ANOVA) for both the services are given in Table 5.

Table 5. Results of Regression Analysis

Performance Metrics	RESTful WS	SOAP basedWS
Confidence level	95%	95%
F ratio	7.85	69
Regression (RN)	2	2
Residuals (RS)	27	27
Critical value of F table [30] i.e F(RN, RS)	3.35	3.35
Throughput and hits/s influence response time	32.09 %	82.4 %

It is observed from Table 5 that the F ratio of SOAP based WS is greater than critical value of F table. Hence F ratio is significant at 0.05. This resembles that there exists a linear relationship in between response time, throughput and hits/s. Therefore, we may reject H_0. This clarifies that the regression equation has 95% chance of being true. Similar results are observed for RESTful based WS. In the regression analysis, the critical value of F table is less than F ratio. Hence, we reject H_0.

The study also suggests that our regression model for SOAP based WS accounts for 82.4% variance on response time and for RESTful WS it is 32.09%. Thus it can be concluded that the throughput and hits/s have an impact on response time for both of the service.

6.3 Chi Square Test and Results

Chi square(χ^2)test is carried out to see whether the frequency distribution fits its expected distribution [31, 35]. It identifies the existence of significant difference between

an observed distribution and a theoretical distribution [35]. The goodness of fit test between expected and observed data can be determined using the chi square equation as [32, 33,34]:

$$\chi^2=\sum (f_o\text{-}fe)^2/f_e \qquad (1)$$

Where f_o is observed frequency and f_e is expected frequency. We assume H_o: the distribution observed fits the distribution expected and H_A: the distribution does not fit the distribution expected.

Table6.χ2Test for SOAP based WS Response Time

Response/s	Observed (fo)	Expected (fe)	fo-fe	$(fo\text{-}fe)^2$	$(fo\text{-}fe)^2/fe$
≤ 10.424	1	3% of 30 = 0.9	0.1	0.01	0.011
>10.424 – ≤ 10.512	1	3% of 30 = 0.9	0.1	0.01	0.011
>10.5128 – ≤ 10.6016	3	10% of 30= 3	0	0	0
>10.6016 – ≤ 10.6904	11	37% of 30= 11.1	0.1	0.01	0.09
>10.6904 – ≤ 10.7792	12	40% of 30= 12	0	0	0
>10.7792	2	7% of 30= 2.1	0.1	0.01	0.005
				χ^2	0.117

Table7.χ2 Test for Response Values of Medical WS based on RESTfulArchitecture

Response/s	Observed (f_o)	Expected (f_e)	$f_o\text{-}f_e$	$(f_o\text{-}f_e)^2$	$(f_o\text{-}f_e)^2/f_e$
≤ 1.662	1	3% of 30 = 0.9	0.1	0.01	0.011
>1.662 – ≤ 1.698	5	17% of 30 = 5.1	0.1	0.01	0.011
>1.698 – ≤ 1.734	8	27% of 30= 8.1	0.1	0.01	0.011
>1.734 – ≤ 1.77	11	37% of 30= 11.1	0.1	0.01	0.011
>1.77 – ≤ 1.806	3	10% of 30= 3	0	0	0
>1.806	2	7% of 30= 2.1	0.1	0.01	0.011
				χ^2	0.055

The degree of freedom (DF) is calculated as 5. It is observed that the critical χ^2 value is 11.0705 for DF 5 at 0.05 confidence level.

It is observed that the calculated χ^2 value for both the WS is less than critical χ^2 value i.e. for SOAP based WS it is 0.117<11.0705 and for RESTful WS it is 0.055<11.0705. Hence we accept the H_0 that the data fits the data distribution expected.

7. Results and Discussion

The investigation reveals that the response time of RESTful architecture is much better than the response time of SOAP based WS. Table 8 presents comparative experimental results of RESTful and SOAP based WS.

Table 8. Comparison of Experimental Results (RT: Response Time, s; TP: Throughput, bytes/s; HT: Hits, s)

		RESTful WS		**SOAP based WS**	
Histogram		Normal with slightly leftskewed		Normal	
Quantile plot		Linear		Linear	
Normal probability pot		Linear		Linear	
VUs	Recorded Parameter	Average	Connection refusal in %	Average	Connection refusal in %
100	RT TP HT	2.123 2226 0.95	0	10.02 2489.27 0.681	0
150	RT TP HT	2.258 2912.07 1.272	0	10.412 4726.324 0.953	0
200	RT TP HT	2.502 3840 1.645	0	10.424 6934.176 1.181	0
250	RT TP HT	2.794 4659.759 1.939	0	11.564 8334.323 1.455	0
350	RT TP HT	2.296 14329.717 4.724	0	11.709 11015.608 1.966	0
500	RT TP HT	1.847 14809.701 3.682	0	11.991 14135.344 2.638	0
600	RT TP HT	2.930 10438.245 4.211	0	10.358 9425.305 3.146	0
700	RT TP HT	2.250 11919.066 4.880	0	11.816 16043.586 3.649	0
800	RT TP HT	2.047 14034 5.669	0	11.776 16834.457 11.776	0
1000	RT TP HT	1.778 16071.575 6.673	0	9.935 9853.486 4.990	1
1200	RT TP HT	1.289 18713.023 8.02	0	10.734 11885.978 5.229	14
1500	RT TP HT	1.268 22610.054 9.811	0	11.019 9049.034 2.969	61

			RESTful WS	SOAP based WS
50 VUs 30 samples	ANOVA	Confidence level	95 %	95 %
		F ratio	7.85	69
		Adjusted R^2	32.09	82.4
		Critical value of F ratio	3.35	3.35
	χ^2	Confidence level	95 %	95 %
		Calculated χ^2 square	0.055	0.117
		DF	5	5
		Critical χ^2 value	11.0705	11.0705

It is observed from Table 8 that the performance attribute values such as throughput, response time of MedWS based on RESTful architecture are much less than the performance attributes of SOAP based service. The throughput of tomcat server with SOAP based WS is much higher than RESTful WS. The SOAP based WS is stable up to 800 VUs without any error but gives low performance at 1500 virtual user with 61% connection refusal. In case of RESTful WS it is observed that the service is error free and stable up to 1500 VU. Hence we can conclude that RESTful WS puts lower overhead and is more efficient than SOAP based WS.

The statistical analysis on the recorded performance metrics of the service shows that SOAP based and RESTful WS are scalable and stable.

It is observed from Table 8 that, the response time, throughput and hits/s for various stress level increases or decreases suddenly. This may be due to partially releasing of server side garbage collected heap which increases server stress.

The SOAP based WS consumes WSDL file of the service provider and processes the XML messages for its communications where as the RESTful WS uses HTTP URI of the resource available over internet and works as like normal HTTP request and response methodology. This may be reason for which the response time of RESTful WS is much less than the SOAP based WS.

The statistical investigation on both the WS predicts that the throughput and hits/s have combined effects on response time. The multiple linear regression analysis for both the WS reveals that the F ratio is significant at 0.05. It is an evidence for linear relationship in between hits/s, throughput and response time of WS.

8. Conclusions

From our overall evaluation on performance testing it can be concluded that both the WS based on SOAP and RESTful architecture are scalable and stable. The statistical analysis of the recorded data shows that the observed parameters are similar to the expected parameters and the data distributions for both the WS are normal. Table 8 gives comparative results of response time, throughput, and hits/s between RESTful and SOAP based WS. From the table we can conclude that RESTful architecture based WS has faster response time than SOAP based WS. For server machines, both architectures will be suitable to be implemented, but for handheld mobile devices that comparatively contains lower hardware resources, WS implementation using RESTful architecture will be preferable than SOAP based WS. Since, RESTful WS has fewer throughputs; it reduces the overheads of overall performance of WS.The experimental results above will give researchers as well as software industrial practitioners an idea about the WS performance and the other metrics that influences the overall performance of the services.

Acknowledgements

The authors are thankful to the All India Council of Technical Education (AICTE), Govt. of India for financial support towards the work (F.No. 8023/BOR/RID/RPS (NER)-84/2010-2011 31st March 2011).

References

[1] I. Siddavatam, J. Gadge, "Comprehensive test mechanism to detect attack on Web Services", Proc. IEEE Int. Conf. on Networks (ICON), **(2008)**; India.
[2] D. Chenthati, H. Mohanty, A. Damodaram, "RDBMS for Service Repository and Matchmaking, ISMS", 2nd Int. Conf.**(2011)**.
[3] P.P.W. Chan, M.R. Lyu, "Dynamic Web Service Composition: A New Approach in Building Reliable Web Service", AINA, 22nd Int. Conf., **(2008)**.
[4] R. Fielding, "Architectural Styles and the Design of Network-based Software architectures", PhD Dissertation, University of California, Irvine, California, USA, **(2000)**.
[5] R.T. Fielding, R.N. Taylor, "Principled design of the modern Web architecture", ACM Transactions on Internet Technology, **(2002)**, pp. 115–150.
[6] C.-J. Su, C. Chang-Yu, "Enabling successful Collaboration 2. 0: A REST -based WebService and Web2.0 technology oriented information platform for collaborative product development", Computers in Industry, **(2012)**.
[7] Available at: http://en.wikipedia.org/wiki/SOAP
[8] Available at: http://www.w3schools.com/wsdl/wsdl_documents.asp
[9] J.Meng, S. Mei, Z. Yan, "RESTful Web Services: A Solution for Distributed Data Integration", International Conference on Computational Intelligence and Software Engineering, CISE, IEEE, **(2009)**.
[10] J. Cox, D. Harvey, D. Ramsbrock, "SOAP vs. REST For Mobile Services", **(2014)** from http://blogs.captechconsulting.com/blog/jack-cox/soap-vs-rest-mobile-services
[11] W3C Working Group, "Web Services Architecture", accessed May **(2011)**from http://www.w3.org/TR/ws-arch/#introduction
[12] F. Aijaz, S. Ali, M. Chaudhary, B. Walke, "Enabling High Performance Mobile Web Services Provisioning", Proceedings of the IEEE 70th Vehicular Technology Conference Fall, IEEE, **(2009)**;Alaska-USA.
[13] L. Richardson, S. Ruby, RESTful Web Services, First Edition, O'Reilly Media, **(2007)**.
[14] H. Hamad, M. Saad, R. Abed, "Performance Evaluation of RESTful Web Services for Mobile Devices", International Arab Journal of e-Technology, vol. 1, no. 3, **(2010)**.
[15] M. B. Juric, I.Rozman, B.Brumen, M.Colnaric, M.Hericko, "Comparison of performance of Web services, WS-Security, RMI, and RMI–SSL", The Journal of Systems and Software, vol. 79, **(2006)**, pp. 689-700.
[16] A. El Saddik, "Performance measurement of Web Service based application, IEEE Transactions on Instrumentation and Measurement, **(2006)**.
[17] B.Upadhyaya, Z.Ying , X.Hua,J. Ng, A. Lau, "Migration of SOAP-based Services to RESTful Services", 13th IEEE International Symposium on Web Systems Evolution (WSE), **(2011)**.
[18] K.E. Mohamed, D.Wijesekera, "Performance Analysis of Web Services on Mobile Devices", The 9th International Conference on Mobile Web Information Systems, Procedia Computer Science 10, **(2012)**.
[19] R. Ramos de Oliveira, R.V. Vieira Sanchez, J.C.Estrella, R. Pontin de Mattos Fortes, V. Brusamolin, "Comparative Evaluation of the Maintainability of RESTful and SOAP-WSDL Web Services", 2013 7thIEEE International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), **(2013)**.
[20] P. Markey, G. Clynch, "A performance analysis of WS-* (SOAP) and RESTful Web Services for Implementing Service and Resource Orientated Architectures", The 12th Information Technology and Telecommunications (IT&T) Conference, Athlone IT, **(2013)**.
[21] Drug Index, Passi Publications, India, January-March, **(2012)**.
[22] M. Kalita, S. Khanikar, T. Bezboruah, "Investigation on performance testing and evaluation of PReWebN: a JAVA technique for implementing web application", IETSoftw., vol. 5, no. 5, **(2011)**, pp. 434-444.
[23] M. Kalita, T. Bezboruah, "Investigation on performance testing and evaluation of PReWebD: a .NET technique for implementing web application", IETSoftw., vol. 5, no. 4, **(2011)**, pp. 357-365.
[24] M. Kalita, T. Bezboruah, "Investigation on implementation of web applications with different techniques", IETSoftw., vol. 6, no. 6, **(2012)**, pp. 474-478.
[25] A. Bora, M.K. Bhuyan, T. Bezboruah, "Investigations on Hierarchical Web service based on Java Technique", Proc. World Congress on Engineering (WCE), **(2013)**; London, U.K.
[26] "Application-testing tool: Mercury LoadRunner 8.0", Available at: http://www.pcquest.com/pcquest/news/183659/application-testing-tool-mercury-loadrunner-80.

[27] R. Brunner, D. Govoni, J. Weber, F. Cohen, F. Curbera, S. Haines, "Java Web Services Unleashed", Sams Indianapolis publisher, **(2002)**; IN, USA.
[28] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, J. Schiller, "Performance Considerations for Mobile Web Services", Computer Communications Journal, vol. 27, no. 11, **(2004)**, pp. 1097-1105.
[29] A. Bogardi-Meszoly, Z. Szitas, T. Levendovszky, H. Charaf, "Investigating factors influencing the response time in ASP.NET web applications", LNCS, 3746, **(2005)**, pp. 223-233.
[30] Available at: http://homepages.wmich.edu/~hillenbr/619/AnovaTable.pdf
[31] Available at: http://fsweb.bainbridge.edu/dbyrd/statistics/goodnessfit.htm
[32] I. Levin, S. Richard, D. Rubin, "Statistics for management", Pearson education, Inc., South Asia, **(2009)**.
[33] Available at: http://www.statisticslectures.com/tables/chisquaretable/
[34] A. Bora, T.Bezboruah, "Testing and Evaluation of a Hierarchical SOAP based Medical Web Service", International Journal of Database Theory Application, vol. 7, no. 5, **(2014)**.

Authors

Abhijit Bora, he is a Research Scholar, Department of Electronics and Communication Technology, Gauhati University, India received Master of Computer Applications (MCA) degree from Jorhat Engineering College (Under Dibrugarh University), India in 2008. His research interests include web service, web security and software engineering.

TulshiBezboruah, he received the B.Sc. degree in physics with electronics from the University of Dibrugarh, Dibrugarh, India, in 1990, and the M.Sc. and Ph.D. degrees in electronics and radio physics from the University of Gauhati, Guwahati, India, in 1993 and 1999, respectively. In 2000, he joined in the Department of Electronics Science, Gauhati University, as a Lecturer. He is currently the Professor & Head, Department of Electronics and Communication Technology, Gauhati University. His current research interests include instrumentation and control, distributed computing, and computer networks. Prof. Bezboruah is a Senior Member of the IEEE, Member of IEEE Geoscience and Remote Sensing Society as well as an Associate Member of the International Center for Theoretical Physics, Trieste, Italy.

